Bankers' Selection Committee Secretariat (BSCS) · Job ID-25106

Assistant Programmer Detailed Preparation

Assistant Programmer / Senior Officer (AP) · Grade 9 · 33 posts

Full solutions of 6 past AP papers, incl. 31.10.2026

Programming problem bank

SQL and DSA hands-on

Ready theory answers

150 MCQs

Math · English · GK

Model tests

Exam: 31 October 2026, Saturday · 9:30 AM – 12:30 PM · Tejgaon Government High School, Dhaka

Contents and Exam Analysis

Chapter	Topic	Why	Page
0	Exam facts, past AP paper analysis, 3-day routine	What comes, what first	2
1	Solved past papers: Combined 5 Bank 2023, Bangladesh Bank 2023, Combined 5 Bank 2024	BIBM problem-based questions	6
2	Solved past papers: BSCS 03.01.2026, BSCS 17.01.2026, BSCS 2020, **BSCS 31.10.2026 (Job ID-25106, complete)**, **Screening MCQ 2026 Set-3 (100 MCQs)**	Recent theory-based questions	29
3	Programming problem bank + output tracing	2–3 code questions in every paper	100
4	SQL query bank and DSA hands-on (heap, hash, graph, sort, paging)	SQL in every paper; DSA in almost all	126
5	10-mark theory: ready answers	2026 papers are mostly "Explain..."	147
6	AP MCQ bank (150)	MCQ 100 marks, −0.25	169
7	Math, English Language Proficiency and GK	50 marks written, 25 marks MCQ	189
8	Model tests (2 sets) and last-night checklist	Practice against the clock	198

0.1 What is confirmed about this exam (Circular No. 340/2026)

Item	Information
Post	Assistant Programmer / Senior Officer (Assistant Programmer), Grade 9, Job ID-25106 — 33 posts in 5 banks and 2 financial institutions under BSCS
Candidates	2,482 (Roll 60001–63310) — about 75 per post
Time	31/10/2026 Saturday, 9:30 AM – 12:30 PM (MCQ 1 hour + written 2 hours, same sitting)
Center	Tejgaon Government High School, Tejgaon, Dhaka; **no entry after 9:15**
MCQ (100)	Subjects Related 75 · Mathematics 15 · General Knowledge 10 · −0.25 for each wrong answer
Written (200)	Subjects Related 150 · Mathematics 20 · English Language Proficiency 30
What to carry	Admit card only (1 copy; a duplicate can be downloaded from erecruitment.bb.org.bd). Phone, calculator, watch, bag, card, paper are banned. Both ears must be visible.

Not confirmed: the question setter

The circular does not name one. In 2023–2026, all bank IT exams with the "1 hour MCQ + 2 hours written in one sitting" format appear in compilations under BIBM's name (latest AP 17.01.2026: "BIBM/CTI"). So **BIBM is the most likely setter** — this is a guess. The past questions are compiled from candidates' memory, so the wording may not be exact.

0.2 Past AP papers — at a glance

Code	Exam	Setter	Written structure
A	Combined 5 Bank AP, 03.02.2023	BIBM	MCQ 80 (IT 75 + Math/GK 25); written IT 150 + Math 20 + Focus 10 + Translation 20
B	Bangladesh Bank AP, 03.02.2023	BIBM	9 IT questions (150) + 4 Math (20) + Focus 10 + Translation 20
C	Combined 5 Bank (10198) AP, 09.02.2024	BIBM	10 IT questions (150) + 4 Math (20) + Focus 20 + Letter 10
D	BSCS AP / Officer (IT), 03.01.2026	—	Non-tech (Bangla essay, English essay, 2 translations, 10 GK) + 15 IT questions × 10
E	BSCS Various Bank AP, 17.01.2026	BIBM/CTI	15 IT questions × 10 + 4 Math + English (10 grammar, paragraph, sentence making)
F	BSCS AP (Job ID-25106), 31.10.2026 — **this circular's exam**	—	15 IT questions × 10 (4 on modern AI) + 4 Math × 5 + English 30 (10 grammar items + 20-mark essay) — fully solved in Section 2.4
G	Screening (preliminary) MCQ test 2026, Set-3 (header appears to read Payra Port)	—	MCQ only: 100 questions, 1 hour, −0.50 per wrong answer — IT 33, Electrical 7, English 20, Bangla 20, GK 19, Math 1; fully solved in Section 2.5

The English part of E is the closest sample to this circular's "English Language Proficiency 30" — a full solution is in Chapter 7.

0.3 How often each topic has appeared (6 AP papers)

Topic	A	B	C	D	E	F	Examples
DBMS / SQL	✔	✔	✔	✔	✔	✔	M:N join query, romantic comedies, salary > avg, relationship types, Book–Borrower, ACID
Software Eng. / UML	✔	✔	✔	✔	✔	✔	Class diagram (Book, Token-ring), Function Point, Waterfall vs Agile, SDLC, Git, ride-sharing UML
DSA	✔	✔	✔	—	✔	✔	Heap, hashing, adjacency, stack vs queue, binary search, 2-stack queue, bubble vs merge, BST traversals
Writing programs	✔	✔	✔	—	✔	—	Smallest in array, LCM, prime 1..n, series, Floyd triangle, flowchart

Security	✔	✔	—	✔	—	✔	TLS private key, DH MitM, CIA, hashing vs encryption, permission precedence, SQL injection + password hashing
OOP	—	✔	✔	—	✔	✔	Overloading/overriding/hiding, polymorphism types, inheritance types, fare polymorphism
OS	✔	—	—	✔	✔	✔	LRU page fault, boot process, process vs thread, producer–consumer, deadlock
Networking / Data comm	✔	—	—	✔	✔	✔	CRC, OSI, TCP vs UDP, OSPF, checksum, HTTP vs HTTPS
Digital logic	—	—	—	✔	✔	—	Alarm truth table, NAND universal
Cloud / Virtualization / RAID	—	—	—	✔	—	✔	Service models, hypervisor, VM vs container, RAID 0/5/10, backup strategy
AI / ML	—	—	—	—	✔	✔	ML paradigms, A* vs BFS, NLP, LLM, agentic AI + MCP, RAG

Four big lessons

1. **SQL and Software Engineering appeared in every AP paper** — you cannot skip the SQL section of Chapter 4 or the UML class diagram.
2. **The style has changed:** 2023–24 questions were calculation/code based (heap trace, hash table, programs); 2026 is almost all "Explain / Differentiate" (15 × 10). So you need Chapters 3–4 (hands-on) *and* Chapter 5 (ready theory).
3. **AI is now a core topic:** paper F (31.10.2026) had four AI questions — NLP, Generative AI/LLM, agentic AI with MCP, and RAG — plus calculation questions on server capacity and cache AMAT. Revise Section 2.4 and the Cloud/AI part of Chapter 5 together.
4. **Banking context:** "new employee account", "permission precedence", "RAID for bank storage" — give banking examples in your answers.

0.4 3-day routine (following this book)

Day	Time	What to do
Day 1 — Wed, 28 Oct	First part	All of Chapter 1 (A, B, C) — try each question yourself first, then check the solution
	Second part	Chapter 3: write the first 15 programs on paper + the output tracing part
	Night	Chapter 6: MCQs on Programming, DSA and DBMS — make a list of mistakes
Day 2 — Thu, 29 Oct	Morning	Chapter 2 (D, E, 2020) — especially E, as it is the most recent AP
	Noon–afternoon	Chapter 4: 30 SQL queries (by hand) + heap/hash/sort/paging trace
	Night	Chapter 5: OOP, DBMS, OS, Security parts; remaining MCQs of Chapter 6

Day 3 — Fri, 30 Oct	Morning	Rest of Chapter 5 (Networking, SE, Cloud/AI, Digital) + Chapter 7 (Math, English, GK)
	9:30–12:30	**Model Test 1 (Chapter 8)** — at the real exam time, with a clock; then mark it (including −0.25)
	Afternoon–night	Written part of Model Test 2 (outline only) + mistake list + last-night checklist of Chapter 8; sleep by 11 PM
Sat, 31 Oct	Morning	Nothing new. Be at the center by 8:30. Carry only the admit card.

0.5 Time and marks strategy in the hall

Part	Time	Strategy
MCQ 100	60 minutes	Round 1: the sure ones (~35 minutes); Round 2: answer if you can eliminate one option, skip if not; last 5 minutes check the OMR
Written IT 150	~90 minutes	If there are 15 questions, ~6 minutes each; known questions first; for "Differentiate", a table of 5–6 rows; in code, short comments and a sample I/O
Math 20	~12 minutes	Show steps — marks come for steps even with a wrong answer
English 30	~18 minutes	Grammar first (quick marks), then paragraph/letter; short, correct sentences

−0.25 negative marking: the maths

Blind guess (4 options): ¼ − ¾ × 0.25 = +0.06 · If you can eliminate one: ⅓ − ⅔ × 0.25 = +0.17 · If you can eliminate two: ½ − ½ × 0.25 = +0.375. **Decision:** answer if you can surely eliminate at least one option.

Chapter 1 — Solved Past Questions: 2023 and 2024

This chapter gives full model answers to **all technical questions** of three BIBM Assistant Programmer written exams. Solutions for maths, English, translation and Focus Writing are in separate chapters. The questions were compiled from candidates' memory, so some wording was incomplete. Where an interpretation had to be assumed, it is stated clearly in the answer.

How to use

First read only the question box and write your own answer on paper in 6–8 minutes. Then compare it with the model answer. Practise the step tables for heap, hashing and LRU, and the code, **by hand**.

1.1 Combined 5 Bank — Assistant Programmer (03.02.2023)

Item	Details
Date	03.02.2023
Set by	BIBM
Pattern	80 MCQ (IT 75 + maths and GK 25); written 200 = IT 150 + 4 maths questions (20) + Focus 10 + translation B/E 20
Topics	Heap, SQL join, UML class diagram, Function Point, array function, TLS, CRC, graph representation, browser security, LRU

Question 1 Combined 5 Bank AP 2023

Show the Max Heap after each of the following operations [a–j]. Also show which value is returned whenever the root is extracted: Insert 5, Insert 6, Insert 8, Extract-Root, Insert 4, Insert 11, Extract-Root, Insert 7, Extract-Root, Extract-Root.

Remember the rules

Insert: Place the new value at the end of the array. Then **sift-up**: swap while the parent (index $\lfloor(i-1)/2\rfloor$) is smaller.
Extract-Root: Return the root (maximum value). Move the last element to the root and **sift-down**: swap with the larger child (children: 2i+1, 2i+2).

Step	Operation	What happens	Heap (array)	Returned
a	Insert 5	Heap is empty; 5 becomes the root	[5]	—
b	Insert 6	6 goes to the end; parent 5 < 6, so swap	[6, 5]	—
c	Insert 8	8 goes to index 2; parent 6 < 8, so swap	[8, 5, 6]	—
d	Extract-Root	8 is removed; last element 6 moves to the root; child 5 < 6, so nothing changes	[6, 5]	**8**

e	Insert 4	4 goes to index 2; parent 6 > 4, so it stays	[6, 5, 4]	—
f	Insert 11	11 goes to index 3; swap with parent 5, then swap with root 6	[11, 6, 4, 5]	—
g	Extract-Root	11 is removed; 5 moves to the root; swap with the larger child 6	[6, 5, 4]	**11**
h	Insert 7	7 goes to index 3; swap with parent 5, then swap with root 6	[7, 6, 4, 5]	—
i	Extract-Root	7 is removed; 5 moves to the root; swap with the larger child 6	[6, 5, 4]	**7**
j	Extract-Root	6 is removed; 4 moves to the root; child 5 > 4, so swap	[5, 4]	**6**

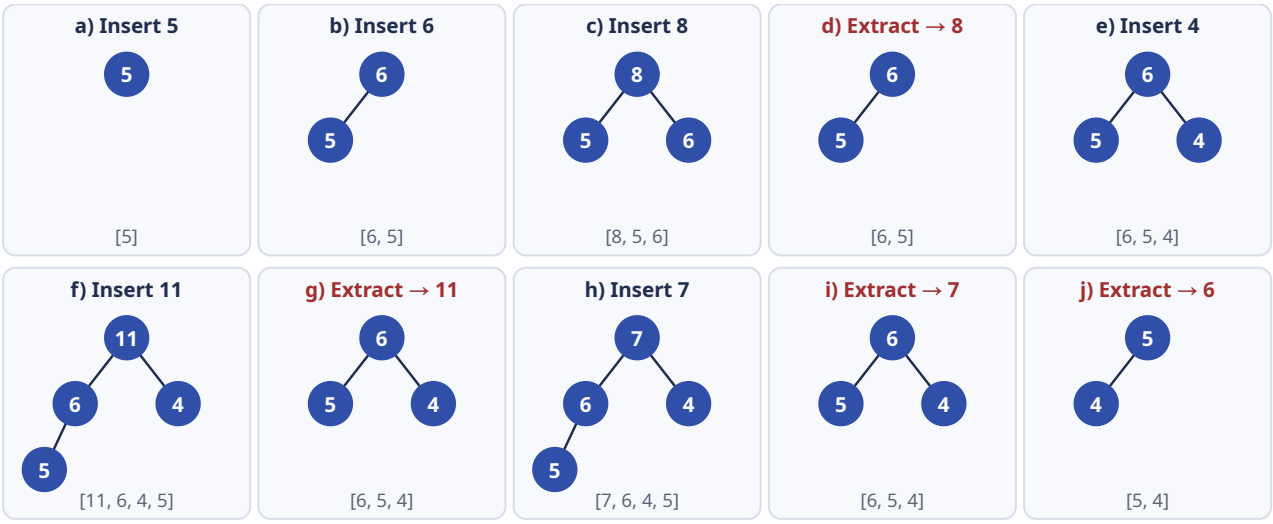


Figure 1.1: Max Heap after each operation (red title = Extract, value after the arrow = returned value)

Answer: The Extract-Root operations return **8, 11, 7, 6** in order. The heap finally holds [5, 4].

Question 2 Combined 5 Bank AP 2023

A relational database has five tables: S(sid, A), T(tid, B), U(uid, C), R(sid, tid, D), Q(tid, uid, E). Primary keys are sid, tid, uid, (sid, tid) and (tid, uid) respectively. R implements a many-to-many relationship between S and T, and Q implements a many-to-many relationship between T and U. Write an SQL query that returns all records of the form (sid, uid), where sid is the key of an S-record, uid is the key of a U-record, and the two records are related through R and Q. Use SELECT, not SELECT DISTINCT.

Logic: An S-record and a U-record are related only when a common T-record (tid) exists such that R has (sid, tid) and Q has (tid, uid). So join R and Q on `tid`.

```
-- Short form (R.sid and Q.uid are foreign keys, so joining S and U is optional)
SELECT R.sid, Q.uid
FROM   R
JOIN   Q ON R.tid = Q.tid;

-- Full form (also confirms that the records really exist in S and U)
SELECT S.sid, U.uid
FROM   S
JOIN   R ON S.sid = R.sid
JOIN   Q ON R.tid = Q.tid
JOIN   U ON Q.uid = U.uid;
```

Why not DISTINCT, and what it means

If the same (sid, uid) pair is linked through more than one tid, it appears more than once in the result (for example s1–t1–u1 and s1–t2–u1). The question forbids DISTINCT, so this repetition is acceptable. Mentioning this in one line shows the examiner that you understand join behaviour.

Question 3 Combined 5 Bank AP 2023

Draw a class diagram for a book: a book is composed of several parts, which are composed of several chapters; chapters are composed of sections. A book has a publisher, publication date and an ISBN; a part has a title and a number; a chapter has a title, a number and an abstract; a section has a title and a number.

"Composed of" means **Composition** (filled diamond ◆). The part cannot exist apart from the whole: if the book is deleted, its parts, chapters and sections are deleted too. Multiplicity is **1** on the whole side and **1..*** on the part side ("several").

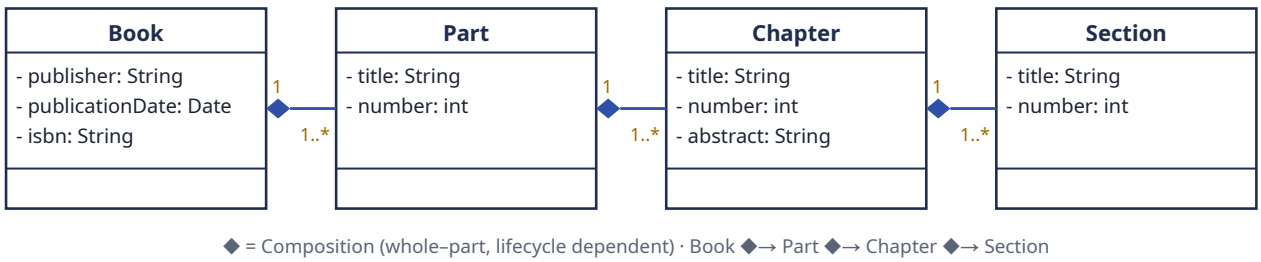


Figure 1.2: Book–Part–Chapter–Section class diagram (composition)

Write this explanation: (1) four classes and the attributes of each; (2) the relationships are composition, because a chapter exists only inside one part; (3) the multiplicity of every relationship is 1 → 1..*. You may also add operations, such as `Book.addPart()` or `Chapter.getSections()`.

Question 4(a) Combined 5 Bank AP 2023

A software project estimates the unadjusted function point (UFP) as 180, the value adjustment factor (VAF) as 0.87, and the performance (productivity) factor as 4. Find the required effort in person-days.

FP = UFP × VAF | Effort = FP ÷ Productivity (FP per person-day)

Step 1: FP = 180 × 0.87 = **156.6**
Step 2: Assume "performance factor 4" means productivity, i.e. **4 FP per person-day**.
Effort = 156.6 ÷ 4 = **39.15 ≈ 40 person-days**

State your assumption clearly

The question does not give the unit of "performance factor". If it means "4 person-days per FP", then Effort = 156.6 × 4 = 626.4 person-days. In the exam, write one line "Assuming productivity = 4 FP/person-day" and give the first answer. VAF normally comes from the formula VAF = 0.65 + 0.01 × ΣF_i; here it is given directly.

Question 4(b) Combined 5 Bank AP 2023

Write a function to find the smallest value in an array. The array is passed to the function as a parameter.

Logic: Treat the first element as min. Loop once over the remaining elements; if one is smaller than min, update min. Time O(n), extra space O(1).

```c
/* C */
#include <stdio.h>

int findMin(int arr[], int n) {
    int min = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] < min)
            min = arr[i];
    return min;
}

int main(void) {
    int a[] = {34, -7, 19, 0, 8};
    int n = sizeof(a) / sizeof(a[0]);
    printf("Smallest = %d\n", findMin(a,
n));
    return 0;
}
/* Output: Smallest = -7 */
```

```java
// Java
public class MinFinder {
    static int findMin(int[] arr) {
        if (arr == null || arr.length ==
0)
            throw new
IllegalArgumentException("empty");
        int min = arr[0];
        for (int i = 1; i < arr.length;
i++)
            if (arr[i] < min) min =
arr[i];
        return min;
    }

    public static void main(String[] args)
{
        int[] a = {34, -7, 19, 0, 8};
        System.out.println("Smallest = " +
findMin(a));
    }
}
// Output: Smallest = -7
```

In C, passing an array actually passes a pointer, so the size (n) must be passed as a separate parameter. In Java, the size comes from `arr.length`.

Question 5(a) Combined 5 Bank AP 2023

Suppose an attacker steals the private key of a website that uses Transport Layer Security (TLS) and remains undetected. What can the attacker do using the private key?

The server proves "I am the real site" using the private key that pairs with the public key in its certificate. If this key is stolen, the attacker can do the following:

1. **Impersonate the server:** Run a fake site with the genuine, valid certificate. If users are redirected there by DNS spoofing, ARP spoofing or a fake Wi-Fi, the browser shows no warning.
2. **Active Man-in-the-Middle:** Sit between the client and the real server and terminate the TLS connection on both sides. The attacker can read passwords, OTPs and card numbers, and can change the amount or destination account of a transaction.
3. **Decrypt recorded traffic (passive):** If the server uses **RSA key exchange** (a non-forward-secret cipher suite of TLS 1.2), the session key comes from the premaster secret sent by the client, encrypted with that public key. So previously recorded sessions and all future sessions can be decrypted.
   If **(EC)DHE** is used (Forward Secrecy, mandatory in TLS 1.3), the private key is used only for signing in the handshake. Then past recorded traffic cannot be decrypted, but an active MitM (point 2) is still possible.
4. **Sign in the server's name:** If the same key was also used to sign something else (such as code or messages), the attacker can forge signatures in the server's name.

Prevention and remedy	Why
Revoke the certificate immediately (CRL/OCSP) and generate a new key pair	The stolen key can no longer gain trust
Allow only ECDHE/TLS 1.3	Ensures forward secrecy
Keep the private key in an HSM	The key cannot be extracted from the HSM
Certificate Transparency monitoring	Fake certificates get detected
Short-lived certificates, regular key rotation	Even if stolen, the damage window is short

Question 5(b) Combined 5 Bank AP 2023

In cyclic redundancy check (CRC), the generator chosen is 11101010111. What is the probability of detecting a burst error of length 10?

The generator has 11 bits, so its degree is **r = 11 − 1 = 10** (the CRC remainder is 10 bits).

Burst length L	Detection probability
$L \le r$	100% (all detected)
$L = r + 1$	$1 - (1/2)^{r-1}$
$L > r + 1$	$1 - (1/2)^{r}$

Here L = 10 and r = 10, so $L \le r$. The highest and lowest (x^0) bits of the generator are both 1, so **every burst error is detected; detection probability = 1 (100%)**.

For comparison: if L = 11, probability = $1 - 1/2^9 \approx 0.998$; if $L \ge 12$, probability = $1 - 1/2^{10} \approx 0.999$.

Question 6 Combined 5 Bank AP 2023

Consider the two standard representations of a directed graph: adjacency list and adjacency matrix. Give one problem that can be solved more efficiently with the adjacency-list representation than with the adjacency-matrix representation, and another problem that can be solved more efficiently with the adjacency matrix than with the adjacency list.

Aspect	Adjacency List	Adjacency Matrix
Space	O(V + E): good for sparse graphs	$O(V^2)$
Find all neighbours of u	O(out-degree(u))	O(V), must scan the whole row
Is edge (u, v) present?	O(out-degree(u))	**O(1)**
BFS / DFS	**O(V + E)**	$O(V^2)$
Find in-degree	O(V + E)	O(V), one column

Faster with a list: In a sparse graph of 1 million nodes and 5 million edges (such as a bank customer transaction network), finding all reachable nodes by BFS/DFS, or counting the out-degree of every node. With a list this takes O(V + E); with a matrix, $O(V^2)$.
Faster with a matrix: Repeatedly asking "is there a direct edge from u to v?" (O(1) lookup), Floyd–Warshall all-pairs shortest path on a dense graph, or checking in O(V) time whether a vertex is a **universal sink** (in-degree V−1, out-degree 0).

Question 7(a) Combined 5 Bank AP 2023

Suppose A and B are two frames in a browser, loaded from different origins. Why is it a reasonable security policy to allow A to navigate B to another origin based only on whether the display area of A contains that of B, i.e., A already controls that area?

This question is about the browser's **frame navigation policy**. Current browsers follow the **Descendant Policy**: a frame may navigate only its own descendant frames to another URL.

1. **Pixel delegation argument:** If B is inside the display area of A, that part of the screen was given by A itself. A could already show its own content there, remove B, or draw something over it. So allowing A to navigate B to another origin gives A **no new power** and does not reduce security.
2. **Same-Origin Policy stays intact:** By navigating, A **cannot read** B's DOM, cookies or data; it only decides what loads in B's place. So confidentiality is not broken.
3. **Risk of a permissive policy:** If any frame could navigate any frame (permissive policy), an attacker frame could send a login or payment frame inside another site to a fake page (frame hijacking / Guninski attack). The user would see the address of the real site, but the data would go to the attacker.
4. **Hence the limit:** "Whoever owns the screen controls it." A can change only frames inside its own area. It cannot change frames of other origins beside or above it.

Question 7(b) Combined 5 Bank AP 2023

Consider the page reference string 1 2 3 4 2 3 4 1 2 1 1 3 1 4 with three frames. Using LRU, find the number of page faults.

In LRU (Least Recently Used), on a page fault the page that has not been used for the longest time is replaced. F = fault, H = hit.

Ref	1	2	3	4	2	3	4	1	2	1	1	3	1	4
Frame 1	1	1	1	**4**	4	4	4	4	4	4	4	**3**	3	3
Frame 2	–	2	2	2	2	2	2	**1**	1	1	1	1	1	1
Frame 3	–	–	3	3	3	3	3	3	**2**	2	2	2	2	**4**
F/H	F	F	F	F	H	H	H	F	F	H	H	F	H	F

Replacement decisions:

- At ref 4, 1 is the oldest, so 1 is replaced.
- At ref 1 (8th), the order is 2, 3, 4 (2 is the oldest), so 2 is replaced.
- At ref 2 (9th), the order is 3, 4, 1, so 3 is replaced.
- At ref 3 (12th), the order is 4, 2, 1, so 4 is replaced.
- At ref 4 (14th), the order is 2, 3, 1, so 2 is replaced.

Answer: Page faults = 8, hits = 6; fault rate = 8/14 ≈ 57%.

1.2 Bangladesh Bank — Assistant Programmer (03.02.2023)

Item	Details
Date	03.02.2023 (1 hour MCQ + 2 hours written)
Set by	BIBM
Pattern	80 MCQ; written = 9 subjective questions (150) + 4 maths questions (20) + Focus (10) + translation B/E (20)
Topics	Hashing, graph representation, LCM program, SQL, UML, overriding/hiding, CIA and E2E encryption, Diffie-Hellman MitM

Question 1 BB AP 2023

Consider a hash table of size 13 with integer keys. The hash function is h(k) = k mod 13. Insert the keys 10, 3, 5, 6, 16, 17, 19 in the given order using linear probing to resolve collisions. Show all the work.

h(k) = k mod 13 | Linear probing: $h_i(k) = (h(k) + i) \bmod 13$, i = 0, 1, 2, …

Key	h(k) = k mod 13	Probe sequence	Final slot
10	10	10 is empty	**10**
3	3	3 is empty	**3**

5	5	5 is empty	**5**
6	6	6 is empty	**6**
16	3	3 (occupied by 3) → 4 is empty	**4**
17	4	4 (16) → 5 (5) → 6 (6) → 7 is empty	**7**
19	6	6 (6) → 7 (17) → 8 is empty	**8**

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Key	–	–	–	3	16	5	6	17	19	–	10	–	–

Total collisions: 1 for 16, 3 for 17, 2 for 19, total 6. The continuous filled run from index 3 to 8 is an example of **primary clustering**, the main weakness of linear probing. Remedies are quadratic probing or double hashing. Load factor α = 7/13 ≈ 0.54.

> **Question 2** BB AP 2023
>
> *Give an adjacency-list representation for a complete binary tree on 7 vertices. Give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7 as in a binary heap.*

With binary-heap numbering, the children of vertex i are **2i** and **2i+1**, and its parent is ⌊**i/2**⌋. The tree is treated as an undirected graph.

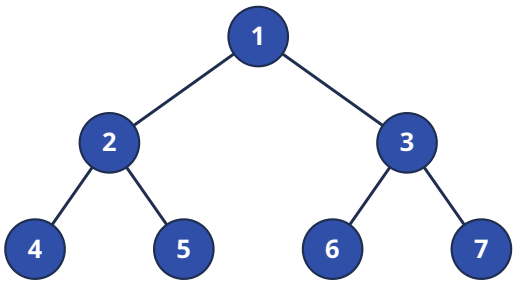


Figure 1.3: Complete binary tree with 7 vertices (heap numbering)

Adjacency List

```
1 → 2 → 3
2 → 1 → 4 → 5
3 → 1 → 6 → 7
4 → 2
5 → 2
6 → 3
7 → 3
```

If treated as directed (parent → child): 1→2,3; 2→4,5; 3→6,7; and the lists of 4–7 are empty.

Adjacency Matrix (symmetric)

	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0
2	1	0	0	1	1	0	0
3	1	0	0	0	0	1	1
4	0	1	0	0	0	0	0
5	0	1	0	0	0	0	0
6	0	0	1	0	0	0	0
7	0	0	1	0	0	0	0

Number of edges = V − 1 = 6. The list has 2 × 6 = 12 entries in total; the matrix has 7 × 7 = 49 cells, of which only 12 are 1. The tree is sparse, so the list saves more space here.

Question 3 BB AP 2023

Given two integers A and B as input, write a program to compute the LCM of A and B.

LCM(A, B) = |A × B| ÷ GCD(A, B) | GCD (Euclid): gcd(a, b) = gcd(b, a mod b), gcd(a, 0) = a

To avoid overflow, compute in the order `(a / gcd) * b`. This divides first, then multiplies.

```c
/* C */
#include <stdio.h>
#include <stdlib.h>

long long gcd(long long a, long long b) {
    while (b != 0) {
        long long t = a % b;
        a = b;
        b = t;
    }
    return a;
}

int main(void) {
    long long a, b;
    printf("Enter A and B: ");
    scanf("%lld %lld", &a, &b);
    a = llabs(a); b = llabs(b);
    if (a == 0 || b == 0) {
        printf("LCM = 0\n");
        return 0;
    }
    long long lcm = (a / gcd(a, b)) * b;
    printf("LCM = %lld\n", lcm);
    return 0;
}
/* Input: 12 18  ->  LCM = 36 */
```

```java
// Java
import java.util.Scanner;

public class LCM {
    static long gcd(long a, long b) {
        while (b != 0) {
            long t = a % b;
            a = b;
            b = t;
        }
        return a;
    }

    public static void main(String[] args)
{
        Scanner sc = new
Scanner(System.in);
        long a = Math.abs(sc.nextLong());
        long b = Math.abs(sc.nextLong());
        long lcm = (a == 0 || b == 0) ? 0
                 : (a / gcd(a, b)) * b;
        System.out.println("LCM = " +
lcm);
    }
}
// Input: 12 18  ->  LCM = 36
```

Trace (12, 18): gcd(12, 18) → gcd(18, 12) → gcd(12, 6) → gcd(6, 0) = 6; LCM = 12/6 × 18 = 36. Time complexity O(log min(A, B)).

Question 4 BB AP 2023

Write an SQL query that returns the number of movies that are romantic comedies. Tables: Movies(mid, title, year), People(pid, name), Genres(gid, genre), ActsIn(pid, mid), HasRole(pid, mid, role), HasGenre(gid, mid). In ActsIn and HasRole, pid is a foreign key into People and mid into Movies; in HasGenre, mid is a foreign key into Movies and gid into Genres.

Logic: "Romantic comedy" means the same movie has two genres: **Romance** and **Comedy**. Each genre is in a separate row of HasGenre, so the two conditions cannot be matched in a single row.

```sql
-- Method 1: GROUP BY + HAVING (cleanest)
SELECT COUNT(*) AS romantic_comedy_count
FROM (
    SELECT hg.mid
    FROM   HasGenre hg
    JOIN   Genres g ON g.gid = hg.gid
    WHERE  g.genre IN ('Romance', 'Comedy')
    GROUP  BY hg.mid
    HAVING COUNT(DISTINCT g.genre) = 2
) AS rc;

-- Method 2: two IN subqueries
SELECT COUNT(*) AS romantic_comedy_count
FROM   Movies m
WHERE  m.mid IN (SELECT hg.mid FROM HasGenre hg JOIN Genres g ON g.gid = hg.gid
                 WHERE g.genre = 'Romance')
  AND  m.mid IN (SELECT hg.mid FROM HasGenre hg JOIN Genres g ON g.gid = hg.gid
                 WHERE g.genre = 'Comedy');
```

Common mistake

Writing `WHERE genre = 'Romance' AND genre = 'Comedy'` always returns **0**, because one row's genre cannot be two values at the same time. Writing `OR` also counts movies that are only romance or only comedy. If the database has a single genre named "Romantic Comedy", the simple `WHERE g.genre = 'Romantic Comedy'` is enough. State this assumption in your answer.

Question 5 BB AP 2023

Draw a class diagram. A token-ring based local-area network (LAN) is a network consisting of nodes, in which network packets are sent around. Every node has a unique name within the network and refers to its next node. Different kinds of nodes exist: workstations are originators of messages; servers and printers are network nodes that can receive messages. Packets contain an originator, a destination and content, and are sent around on a network. A LAN is a circular configuration of nodes.

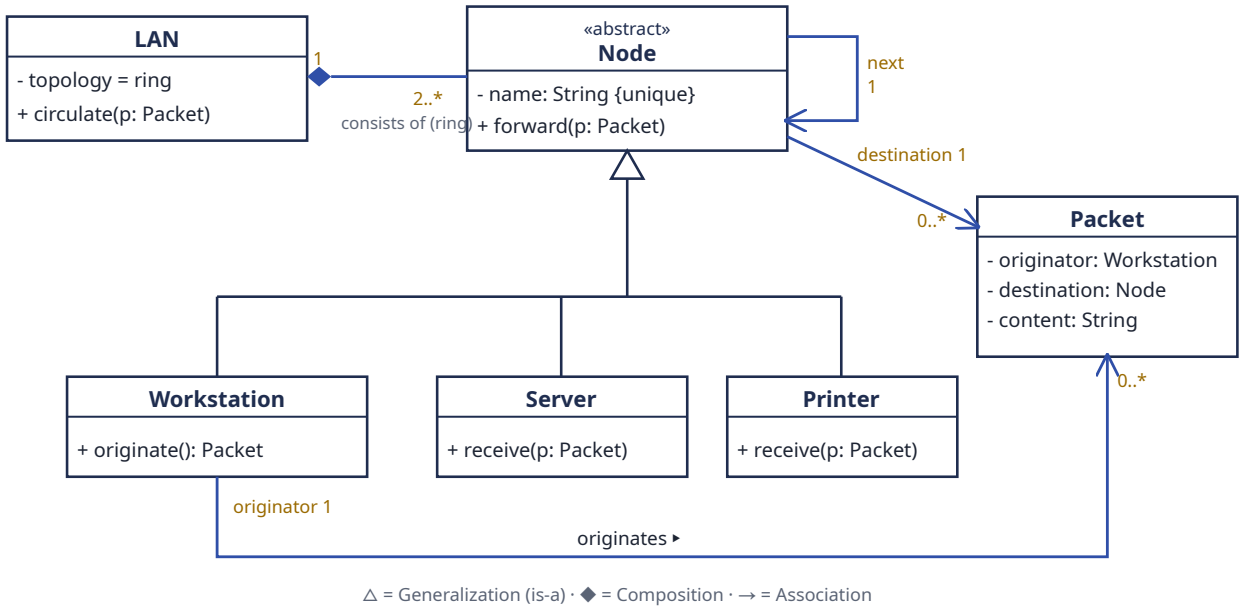


Figure 1.4: Class diagram of a token-ring LAN

Sentence	What it means in the model
"LAN consists of nodes; circular configuration"	LAN ◆— Node, composition, multiplicity 1 → 2..*; the ring is formed by a self-association
"Every node has a unique name and refers to its next node"	Node has `name {unique}`; self-association `next`, multiplicity 1
"Different kinds of nodes"	Node is an abstract superclass; Workstation, Server and Printer are its subclasses (generalization)
"Workstations are originators of messages"	Workstation 1 —originates→ 0..* Packet
"Servers and printers can receive messages"	Server and Printer have a `receive(p: Packet)` operation
"Packets contain originator, destination and content"	Packet attributes or associations: originator (Workstation), destination (Node), content

Question 6 BB AP 2023

Determine the overloaded methods, the overridden methods and the hidden superclass methods in the following code:

```
public class A {
    public void m1() {}
    public void m2() {}
    public void m3(int i) {}
    public void m4(String i) {}
}
public class B extends A {
    public static void m1() {}
    public void m2() {}
    public void m3(int i) {}
    public static void m4(String i) {}
}
```

Concept	Condition	When it is resolved
Overloading	Same name, different parameter list (in the same class or a subclass)	Compile time
Overriding	An **instance** method with the same signature in the subclass; the parent's method is also an instance method	Runtime (dynamic dispatch)
Hiding	**Both** parent and child methods are **static**, with the same signature	Compile time (based on the reference type)

Method	In A	In B	Result
m1()	instance	static	**Compile error**: "m1() in B cannot override m1() in A; overriding method is static". A static method cannot hide an instance method.

m2()	instance	instance, same signature	**Overridden**
m3(int)	instance	instance, same signature	**Overridden**
m4(String)	instance	static	**Compile error** (same reason as m1)

Conclusion: m2 and m3 are **overridden**. m1 and m4 will not compile, so no **hiding** happens here. No method has a different parameter list, so there is no **overloading** either. Also fix two problems in the code: `extends a` must be `extends A`, and one .java file cannot contain two public classes, so put them in two separate files.

When hiding and overloading would occur: If A's m1 were `public static void m1()`, B's static m1() would **hide** it. If B had `public void m3(double d)`, it would **overload** A's m3(int).

```
class A {
    public static void s()   { System.out.println("A.s"); }
    public void m()          { System.out.println("A.m"); }
}
class B extends A {
    public static void s()   { System.out.println("B.s"); }   // hiding
    @Override public void m(){ System.out.println("B.m"); }   // overriding
    public void m(int x)     { System.out.println("B.m(int)"); } // overloading
}
public class Demo {
    public static void main(String[] args) {
        A ref = new B();
        ref.m();   // B.m  : at runtime, uses the object's type (B)
        ref.s();   // A.s  : at compile time, uses the reference's type (A), i.e. hiding
    }
}
```

Question 7 BB AP 2023

A question about the CIA triad: to protect its data, a company decided to add end-to-end encryption. Which layer of the OSI model is suitable for it, considering development time, software maintainability and development cost? Give reasons.

CIA Triad:

- **Confidentiality:** Unauthorized parties cannot read the information.
- **Integrity:** Information cannot be changed without authorization.
- **Availability:** Authorized users get the information when they need it.

End-to-end encryption (E2EE) mainly ensures **confidentiality** and, with a MAC or AEAD, **integrity**.

Answer: Application Layer (Layer 7).

Layer	Example	Is it E2E?	Cost and maintenance
Network (L3)	IPsec	No; it protects host-to-host or gateway-to-gateway, not user-to-user	Needs OS and router configuration; needs experts

Transport (L4)	TLS	No; it is decrypted at every hop (server, proxy, load balancer), so the server sees plaintext	Easy, but does not meet the goal
Application (L7)	Signal Protocol, PGP, libsodium	**Yes**; only the sender's and receiver's apps hold the keys	**Ready-made libraries; no network infrastructure change**

Reasons:

1. **Development time:** Tested libraries (OpenSSL, libsodium, Bouncy Castle) can be used directly in the app code. No need to touch the kernel or network devices.
2. **Maintainability:** The encryption logic stays inside the app. Nothing needs to change if the network, cloud provider or ISP changes, and updates ship like any normal software release.
3. **Development cost:** No cost for special hardware (VPN concentrator) or for configuring every router.
4. **True end-to-end:** Only at this layer can intermediate servers, even the company's own servers, not read the data. So confidentiality holds even if data leaks.

Limitations: it must be implemented separately in each app, and key management (key exchange, backup) must be designed carefully. In practice, TLS is also kept underneath Application-layer E2EE (defense in depth).

> **Question 8** BB AP 2023
>
> *Describe a man-in-the-middle attack on the Diffie-Hellman key exchange protocol in which the adversary generates two public key pairs for the attack.*

Core weakness: Basic Diffie-Hellman has **no authentication**. Nobody verifies that a received public key really belongs to the other party. Public values: prime q and primitive root α.

Attack steps (attacker Darth, as described by Stallings):

1. Darth picks two private keys X_{D1} and X_{D2} and computes the matching public keys: $Y_{D1} = \alpha^{X_{D1}} \bmod q$, $Y_{D2} = \alpha^{X_{D2}} \bmod q$.
2. Alice sends Y_A to Bob. Darth intercepts it, sends Y_{D1} to Bob, and computes $K2 = Y_A^{X_{D2}} \bmod q$.
3. Bob receives Y_{D1} and computes $K1 = Y_{D1}^{X_B} \bmod q$.
4. Bob sends Y_B to Alice. Darth intercepts it, sends Y_{D2} to Alice, and computes $K1 = Y_B^{X_{D1}} \bmod q$.
5. Alice receives Y_{D2} and computes $K2 = Y_{D2}^{X_A} \bmod q$.

Result: Alice and Darth share the same key K2, and Bob and Darth share the same key K1. Alice and Bob think they share a secret key with each other. Darth can decrypt and read or modify every message, then re-encrypt it and forward it.

Numerical example: q = 23, α = 5

Alice: $X_A = 6 \rightarrow Y_A = 5^6 \bmod 23 = \mathbf{8}$ | Bob: $X_B = 15 \rightarrow Y_B = 5^{15} \bmod 23 = \mathbf{19}$

Darth: $X_{D1} = 3 \rightarrow Y_{D1} = 125 \bmod 23 = \mathbf{10}$ | $X_{D2} = 7 \rightarrow Y_{D2} = 5^7 \bmod 23 = \mathbf{17}$

Alice: $K2 = 17^6 \bmod 23 = \mathbf{12}$ | Darth: $K2 = 8^7 \bmod 23 = \mathbf{12}$ ✔

Bob: $K1 = 10^{15} \bmod 23 = \mathbf{5}$ | Darth: $K1 = 19^3 \bmod 23 = \mathbf{5}$ ✔

Without the attack, Alice and Bob would get the same key: $19^6 \bmod 23 = 8^{15} \bmod 23 = 2$.

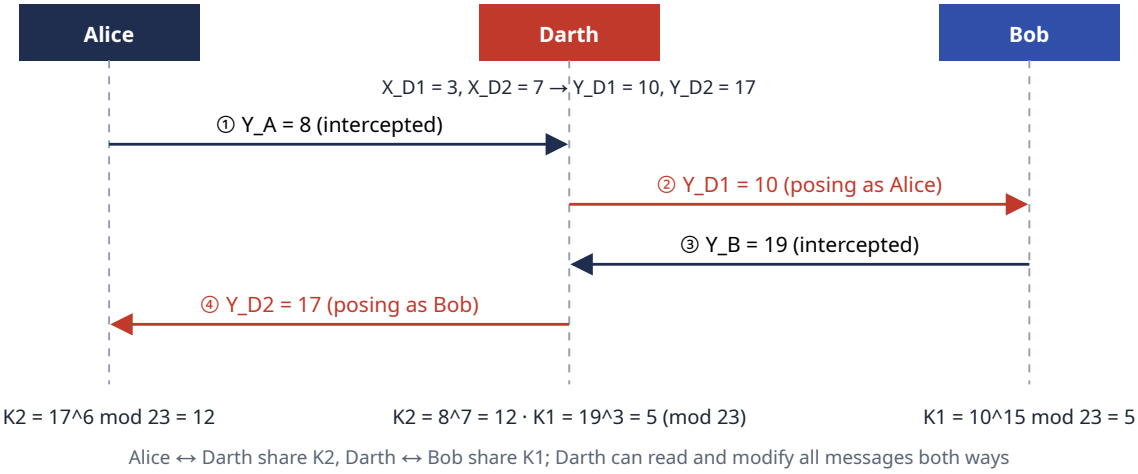


Figure 1.5: MitM attack on Diffie-Hellman using two key pairs

Prevention:

- **Authenticate** the public keys: DH with digital signatures or certificates (Station-to-Station protocol, signed ECDHE in TLS).
- Compare key fingerprints over a separate channel (e.g. Signal's safety number).
- Use a pre-shared key or PKI.

1.3 Combined 5 Bank (Job ID 10198) — Assistant Programmer (09.02.2024)

Item	Details
Banks	Sonali Bank, BKB, Karmasangsthan Bank, ICB, etc.
Date	09.02.2024 (1 hour MCQ + 2 hours written)
Question setter	BIBM
Type	80 MCQ (IT 50 + Math and GK 20); written = 10 subjective (150) + 4 math (20) + Focus (20) + Letter (10)
Feature	**Programming-heavy**: 3 programs to write, 1 output to find, 3 SQL queries, then theory and a question about your own project

Question 1 Combined 5 Bank AP 2024

Write a C/Java program that shows the prime numbers between 1 and n, where n is given by user input.

Logic: x is prime if x ≥ 2 and no number from 2 to √x divides x. Using the loop condition `d * d <= x` avoids the need for a sqrt function. Each number costs O(√x), total O(n√n). For large n, the Sieve of Eratosthenes is faster: O(n log log n).

```c
/* C */
#include <stdio.h>

int isPrime(int x) {
    if (x < 2) return 0;
    for (int d = 2; d * d <= x; d++)
        if (x % d == 0) return 0;
    return 1;
}

int main(void) {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    printf("Primes between 1 and %d:\n",
n);
    for (int i = 2; i <= n; i++)
        if (isPrime(i))
            printf("%d ", i);
    printf("\n");
    return 0;
}
```

```java
// Java
import java.util.Scanner;

public class Primes {
    static boolean isPrime(int x) {
        if (x < 2) return false;
        for (int d = 2; d * d <= x; d++)
            if (x % d == 0) return false;
        return true;
    }

    public static void main(String[] args)
{
        Scanner sc = new
Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt();
        for (int i = 2; i <= n; i++)
            if (isPrime(i))
                System.out.print(i + " ");
        System.out.println();
    }
}
```

Sample: Enter n: 30 → `2 3 5 7 11 13 17 19 23 29`. 1 is not prime, so the loop starts at 2.

Question 2 Combined 5 Bank AP 2024

Write a C/Java program that calculates the sum of the series 1 + 2 + 4 + 7 + 11 + ... up to the n-th term.

Pattern: The differences between adjacent terms are 1, 2, 3, 4, ... So each term is the previous term plus k (k = 1, 2, 3, ...).

k-th term: $t_k = 1 + k(k - 1)/2$ | Sum of n terms: $S_n = n + (n - 1)n(n + 1)/6$

Check: $t_5 = 1 + 5{\cdot}4/2 = 11$ ✔; $S_5 = 5 + 4{\cdot}5{\cdot}6/6 = 5 + 20 = 25 = 1 + 2 + 4 + 7 + 11$ ✔

```c
/* C */
#include <stdio.h>

int main(void) {
    int n;
    long long term = 1, sum = 0;
    printf("Enter n: ");
    scanf("%d", &n);
    for (int k = 1; k <= n; k++) {
        printf("%lld%s", term, k < n ? " + " : "");
        sum += term;
        term += k;      /* next term = term + k */
    }
    printf("\nSum = %lld\n", sum);
    return 0;
}
```

```java
// Java
import java.util.Scanner;

public class SeriesSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt();
        long term = 1, sum = 0;
        for (int k = 1; k <= n; k++) {
            System.out.print(term + (k < n ? " + " : ""));
            sum += term;
            term += k;
        }
        System.out.println("\nSum = " + sum);
        // closed form check:
        long f = n + (long)(n - 1) * n * (n + 1) / 6;
        System.out.println("Formula = " + f);
    }
}
```

Sample: n = 5 → `1 + 2 + 4 + 7 + 11`, `Sum = 25`. For n = 7 the terms go up to 16 and 22, Sum = 63 (by formula 7 + 6·7·8/6 = 63 ✔).

Question 3 Combined 5 Bank AP 2024

Write a program in C/Java to print the following Floyd's triangle (0–1 pattern). N is given by user input. For n = 5:

```
1
01
101
0101
10101
```

Pattern analysis: Row i has i digits. Odd rows start with 1 and even rows start with 0, then 1 and 0 alternate. Both rules combine into one: **if (i + j) is even print 1, if odd print 0** (i and j counted from 1).

```c
/* C */
#include <stdio.h>

int main(void) {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++)
            printf("%d", (i + j) % 2 == 0
? 1 : 0);
        printf("\n");
    }
    return 0;
}
```

```java
// Java
import java.util.Scanner;

public class BinaryFloyd {
    public static void main(String[] args)
{
        Scanner sc = new
Scanner(System.in);
        int n = sc.nextInt();
        for (int i = 1; i <= n; i++) {
            StringBuilder row = new
StringBuilder();
            for (int j = 1; j <= i; j++)
                row.append((i + j) % 2 ==
0 ? 1 : 0);
            System.out.println(row);
        }
    }
}
```

Check: i = 1: j = 1 → 2 is even → "1". i = 2: 3 → 0, 4 → 1 → "01". i = 4: 5, 6, 7, 8 → "0101". Matches the sample in the question exactly ✔. If you want the usual number-based Floyd's triangle (1 / 2 3 / 4 5 6 ...), keep a counter and write `printf("%d ", c++)` each time.

Question 4 Combined 5 Bank AP 2024

Find the output of the following program:

```c
#include <stdio.h>
int fun(int i) {
    if (i <= 0) return;
    printf("%d, ", i--);
    fun(--i);
    printf("%d, ", i);
}
int main() {
    fun(5);
}
```

Remember: `i--` (post-decrement) uses the value first, then decrements. `--i` (pre-decrement) decrements first, then uses the value. So in each call, i decreases by 2 in total. The second printf runs after the recursive call returns, so those values print in **reverse order**.

Call	First printf	Change in i	Recursive call	Second printf after return
fun(5)	"5, "	i-- → 4, --i → 3	fun(3)	"3, " (last)
fun(3)	"3, "	i-- → 2, --i → 1	fun(1)	"1, "
fun(1)	"1, "	i-- → 0, --i → −1	fun(−1)	"−1, "
fun(−1)	—	i ≤ 0, so it returns without printing anything		

Order: going down: 5, 3, 1 → fun(−1) returns → coming back up: −1 (fun(1)), 1 (fun(3)), 3 (fun(5)).

Output: 5, 3, 1, -1, 1, 3,

Subtle point (write it in one line in the answer)

`int fun()` is a non-void function, yet `return;` has no value and there is no return at the end. In C99/C11 this is a constraint violation or a warning, but since the returned value is never used, most compilers (gcc) give a warning and still produce the output above. The "I", "Printf" and "Ø" in the question paper are OCR or typing errors; they are taken as `i`, `printf` and `0`.

Questions 5–6 Combined 5 Bank AP 2024

Given two tables Department and Employee:

```
CREATE TABLE Department (DeptCode INT, DeptName VARCHAR(20), Location VARCHAR(20));
CREATE TABLE Employee   (EmpCode INT, EmpFname VARCHAR(20), Salary INT,
                         Commission INT, DeptCode VARCHAR(20));
```

(a) Select every EmpFname with total salary (total salary = salary + commission). (b) Select EmpFname and Location of employees whose salary is greater than the average salary, ordered by EmpFname. (c) Select EmpFname and the count of employees, ordered by the employee count ascending.

Schema correction (mention in the answer): The type of Employee's DeptCode should be `INT`, the same as Department's DeptCode, so that the join and foreign key work properly. In the original DDL there is no comma after `Commission INT` and there is an extra comma at the end; both are syntax errors.

```
CREATE TABLE Department (
    DeptCode  INT PRIMARY KEY,
    DeptName  VARCHAR(20),
    Location  VARCHAR(20)
);
CREATE TABLE Employee (
    EmpCode    INT PRIMARY KEY,
    EmpFname   VARCHAR(20),
    Salary     INT,
    Commission INT,
    DeptCode   INT REFERENCES Department(DeptCode)
);

-- (a) Total salary; if Commission is NULL, treat it as 0
SELECT EmpFname,
       Salary + COALESCE(Commission, 0) AS TotalSalary
FROM   Employee;

-- (b) Name and Location of those who earn more than the average salary
SELECT   e.EmpFname, d.Location
FROM     Employee e
JOIN     Department d ON e.DeptCode = d.DeptCode
WHERE    e.Salary > (SELECT AVG(Salary) FROM Employee)
ORDER BY e.EmpFname;

-- (c) Number of employees for each EmpFname, in order of count
SELECT   EmpFname, COUNT(EmpCode) AS EmpCount
FROM     Employee
GROUP BY EmpFname
ORDER BY EmpCount ASC;
```

Explanation and alternatives

(a): `NULL + number = NULL`, so without COALESCE (or IFNULL in MySQL, ISNULL in SQL Server), an employee with no commission would show a NULL total salary.
(b): `AVG()` cannot be used directly in WHERE, so a subquery is needed.
(c): The question is ambiguous. If a non-aggregate column (EmpFname) is kept with an aggregate, it must be in GROUP BY. Above, it is taken as "how many employees share the same name". If the intent is the number of employees in each department:

```
SELECT d.DeptName, COUNT(e.EmpCode) AS EmpCount
FROM Department d LEFT JOIN Employee e ON e.DeptCode = d.DeptCode
GROUP BY d.DeptName
ORDER BY EmpCount ASC;
```

And to show the total number of employees beside each name, use a window function: `SELECT EmpFname, COUNT(*) OVER () AS TotalEmp FROM Employee ORDER BY EmpCode;`. Write your own interpretation in one line on the answer sheet.

Question 7 Combined 5 Bank AP 2024

What is the difference between a stack and a queue? Also write two uses of each.

Aspect	Stack	Queue
Principle	**LIFO** (Last In First Out)	**FIFO** (First In First Out)
Ends	One end (top). Both insert and delete happen here	Two ends: insert at rear, delete at front
Operations	push, pop, peek	enqueue, dequeue, front
Pointer	top	front and rear
Complexity	push/pop O(1)	enqueue/dequeue O(1) (in a circular array or linked list)
Variants	—	Circular queue, priority queue, deque
Real-life example	A pile of plates	A line at a bank counter

Uses of a stack

1. Call stack for function calls and recursion, holding return addresses and local variables.
2. Expression evaluation and infix → postfix conversion; checking balanced parentheses.
3. (Extra) Undo/Redo, browser Back button, DFS.

Uses of a queue

1. CPU scheduling (Round Robin ready queue) and printer spooling.
2. BFS traversal; packet buffer in a network router.
3. (Extra) Bank token system, message queues (RabbitMQ, Kafka).

Question 8 Combined 5 Bank AP 2024

What is polymorphism? Describe the different types of polymorphism.

Definition: Polymorphism (poly = many, morph = form) means using the same interface or name to get different behavior from objects of different types. It is one of the four pillars of OOP; the other three are encapsulation, inheritance and abstraction.

Type	Other name	How	Binding
Compile-time	Static polymorphism	Method overloading (same name, different parameters); operator overloading in C++	Early/static binding
Runtime	Dynamic polymorphism	Method overriding + upcasting (parent reference, child object); virtual functions in C++	Late/dynamic binding
Parametric	Generics / Templates	Java's `List<T>`, C++ templates. The same code works with different types	Compile time

```
// Compile-time: overloading
class Calculator {
    int    add(int a, int b)          { return a + b; }
    double add(double a, double b)    { return a + b; }
    int    add(int a, int b, int c)   { return a + b + c; }
}

// Runtime: overriding
abstract class Account {
    abstract double interest(double balance);
}
class Savings extends Account {
    @Override double interest(double b) { return b * 0.05; }
}
class FixedDeposit extends Account {
    @Override double interest(double b) { return b * 0.09; }
}

public class Bank {
    public static void main(String[] args) {
        Calculator c = new Calculator();
        System.out.println(c.add(2, 3));        // 5    (int version)
        System.out.println(c.add(2.5, 3.5));    // 6.0  (double version)

        Account[] list = { new Savings(), new FixedDeposit() };
        for (Account a : list)                    // same call, different behavior
            System.out.println(a.interest(10000)); // 500.0, 900.0
    }
}
```

Benefits: Adding a new account type does not require changing Bank's loop (Open–Closed principle). Code becomes reusable and extensible.

> **Question 9** Combined 5 Bank AP 2024
>
> *Describe the different types of relationships in a DBMS.*

In the ER model, a **relationship** is an association between two or more entities. It is classified in three ways.

(a) By cardinality (mapping)

Type	Meaning	Example	Implementation in tables
One-to-One (1:1)	One record of A relates to at most one of B	Customer — NID; Employee — Desk	UNIQUE foreign key on either side
One-to-Many (1:N)	One record of A relates to many of B	Branch — Accounts; Department — Employees	Foreign key in the "many" side table
Many-to-One (N:1)	1:N seen from the opposite side	Accounts — Branch	Same

Many-to-Many (M:N)	Many on both sides	Customer — Account (joint account); Student — Course	**Junction (bridge) table**, composite PK from the two FKs, like R and Q in Question 2

(b) By degree

- **Unary (recursive):** An entity is related to itself, e.g. Employee *manages* Employee, or Node *next* Node (token ring).
- **Binary:** Two entities, e.g. Customer *holds* Account. This is the most common.
- **Ternary:** Three entities together, e.g. Doctor–Patient–Medicine (*prescribes*), or Supplier–Part–Project.

(c) By participation

- **Total (mandatory):** Every record must take part in the relationship, e.g. every Loan must have a Borrower. In an ER diagram it is shown with a double line.
- **Partial (optional):** Not every record needs to take part, e.g. not every Customer has a Loan.

If "relation" means a table: base relation (stored table), view (virtual relation) and query result (derived relation). Add this in one line at the end of the answer if needed.

Question 10 Combined 5 Bank AP 2024

Describe a real-life software project you have built. What problems or challenges did you face during that time, and how did you solve them?

Strategy

Pick your own thesis, internship or job project and write it in the structure below. A banking-related project is even better. The examiner checks whether you can identify problems and give **technical solutions**. Writing only "it was very hard" earns no marks. For each challenge, name the specific tool or technique.

Model answer (adapt it to your own project):

1. Project overview: "Micro-Loan Management System" — a web application that manages loan applications, approval, installment collection and reports for an NGO or microfinance institution. About 50 field workers and 10 branch managers use it; about 20,000 customer records.

2. Technology and my role: Java (Spring Boot) REST API, MySQL, React frontend, Git and Docker. As backend developer, I built the database design, the installment calculation module and reporting. Work followed Agile (2-week sprints).

Challenge	Cause	Solution
Same installment posted twice (double posting)	On a weak network, field workers pressed Submit repeatedly; concurrent requests	Used transactions (`@Transactional`) and row-level locks; an idempotency key on every payment; disabled the button after a click

Monthly report very slow (about 40 seconds)	Full table scan on large tables; N+1 queries	Analyzed with `EXPLAIN`; composite index on (branch_id, due_date); JOIN FETCH; summary table updated by a nightly batch. Time dropped below 2 seconds
Requirements kept changing	Interest rates and penalty rules differ by branch	Kept the rules in a configuration table instead of hard-coding them; took feedback with a demo at the end of every sprint
Security	Sensitive financial and personal data	Password hashing with bcrypt; role-based access (RBAC); prepared statements to prevent SQL injection; HTTPS; audit log
"It works on my PC but not on the server" in deployment	Environment differences	Docker image, CI pipeline (GitHub Actions) and automated unit and integration tests

3. Result and lessons: A real-time dashboard replaced paper registers, and calculation errors fell to almost zero. Lessons: ensure data integrity (transactions, constraints) from the start, measure performance before optimizing, and keep regular communication with users. As a bank Assistant Programmer, this experience applies directly to building CBS-adjacent apps or internal tools.

Chapter 2 — Solved Past Questions: 2026 (and 2020)

This chapter gives full 10-mark model answers to **only the technical (subject-related) questions** of the two recent AP written exams and the 2020 BSCS Officer (IT) written exam. The Mathematics, English, translation, essay and GK parts are in Chapter 7. The questions were compiled from candidates' memory, so OCR/typing errors have been corrected silently; where a guess was needed, it is stated clearly.

> **Why these three papers matter**
> The 31 October AP (Job ID-25106) circular gives the written exam as: **Subjects Related 150 + Mathematics 20 + English Language Proficiency 30**. The January 2026 AP paper (Paper E) followed exactly this pattern: 15 subject questions, 4 mathematics questions and a separate "English Proficiency" part (grammar, paragraph, sentence making). So prepare with Paper E as the closest model.

2.1 Paper D — BSCS Assistant Programmer / Officer (IT), 03.01.2026

Item	Information
Exam	BSCS (Sonali and other Bank PLCs) — Assistant Programmer / Officer (IT)
Date and time	03/01/2026, 9 am–12 noon
Written	200 marks: Non-tech (Bangla essay, English essay, translation B→E and E→B, GK) + Subject Related Questions 6–20 (15 × 10 = 150)
Note	In the compilation, Q13/15 (new employee account) and Q14/16 (alarm truth table) were printed twice. Here each appears once, arranged in the paper's order (6–20)

Question 6 BSCS AP 03.01.2026

Mention the layers of the OSI Model and the function of each layer. (10)

The **OSI (Open Systems Interconnection) Model** is a 7-layer reference model made by ISO. It divides the work of data exchange between two computers and gives each layer a specific duty. On the sender's side, data moves from top to bottom (each layer adds a header — *encapsulation*). On the receiver's side, it moves from bottom to top (*decapsulation*).

Layer	Name	Main function	PDU	Protocols / Devices
7	Application	Gives network services to the user/application (web, e-mail, file transfer)	Data	HTTP, HTTPS, FTP, SMTP, DNS
6	Presentation	Data translation (syntax), encryption/decryption, compression	Data	TLS/SSL, JPEG, ASCII, MPEG
5	Session	Establishes, maintains and ends sessions; dialog control, synchronization (checkpoints)	Data	NetBIOS, RPC

4	Transport	End-to-end delivery, segmentation/reassembly, port addressing, flow and error control	Segment / Datagram	TCP, UDP
3	Network	Logical (IP) addressing, routing — choosing the best path from source to destination	Packet	IP, ICMP, OSPF; Router
2	Data Link	Framing, physical (MAC) addressing, hop-to-hop error detection (CRC), media access	Frame	Ethernet, PPP; Switch, Bridge
1	Physical	Converts bits into electrical/light/radio signals; cables, connectors, bit rate	Bit	Cable, Hub, Repeater, NIC

Memory trick

Top to bottom: **A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing. Bank example: when a customer logs in to internet banking: HTTPS (L7) → TLS encryption (L6) → session (L5) → TCP port 443 (L4) → IP routing (L3) → Ethernet frame (L2) → bits on fibre/wire (L1).

Question 7 BSCS AP 03.01.2026

Compare TCP and UDP protocols with examples. (10)

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are both Transport layer protocols of OSI. TCP gives reliability; UDP gives speed.

Basis	TCP	UDP
Connection	Connection-oriented (3-way handshake: SYN → SYN-ACK → ACK)	Connectionless — sends directly
Reliability	Reliable — ACK, retransmission, sequence numbers	Unreliable — no delivery guarantee
Order	In-order delivery	Order not guaranteed
Flow/Congestion control	Yes (sliding window, slow start)	No
Header	20–60 bytes	8 bytes
Speed/overhead	Slower, more overhead	Faster, less overhead
Broadcast/Multicast	Not supported	Supported
Examples	HTTP/HTTPS (80/443), FTP (20/21), SMTP (25), SSH (22), internet banking transactions	DNS (53), DHCP (67/68), SNMP (161), VoIP, live video streaming, online games

Conclusion: Use TCP where every byte must arrive correctly (fund transfer, file download). Use UDP where losing a little data is acceptable but delay is not (voice/video calls).

Question 8 BSCS AP 03.01.2026

Explain the difference between compiler and interpreter. (10)

Both convert source code written in a high-level language into a form the machine can understand. A **compiler** translates the whole program at once and produces an executable (object code). An **interpreter** translates each statement one by one and runs it immediately.

Basis	Compiler	Interpreter
Unit of translation	Whole program at once	Line by line (statement by statement)
Output	Separate object/executable file (.exe, .o)	No permanent executable
Execution speed	Fast (compile once, run many times)	Slow (translates every time)
Error reporting	All errors together at the end of compilation	Stops at the first error — easier debugging
Memory	More, to hold object code	Less
Source code needed	Not needed at run time	Needed every time
Examples	C, C++, Go, Rust	Python, JavaScript, Ruby, PHP

Hybrid approach: In Java, `javac` compiles the source to bytecode. Then the JVM interprets it or converts it to native code with a JIT (Just-In-Time) compiler. Python also first creates .pyc bytecode. Compiler phases: Lexical analysis → Syntax analysis → Semantic analysis → Intermediate code → Optimization → Code generation.

> **Question 9** BSCS AP 03.01.2026
>
> *What is cloud computing? Mention its service models. (10)*

Definition (NIST): Cloud computing is a model that gives on-demand network access to a shared pool of configurable computing resources (servers, storage, networks, applications, services). These can be provisioned and released quickly with minimal management effort.

5 essential characteristics: On-demand self-service, Broad network access, Resource pooling (multi-tenancy), Rapid elasticity, Measured service (pay-as-you-go).

Service Model	What the provider gives	What the customer controls	Examples
IaaS (Infrastructure as a Service)	Virtual machine, storage, network	OS, runtime, applications, data	AWS EC2, Azure VM, Google Compute Engine
PaaS (Platform as a Service)	IaaS + OS + runtime + middleware + database	Only applications and data	Google App Engine, Heroku, Azure App Service
SaaS (Software as a Service)	Fully built software	Only usage and some settings	Gmail, Microsoft 365, Salesforce, Zoom

Deployment models: Public, Private, Hybrid, Community cloud. For a bank, private/hybrid cloud is usually used for sensitive customer data, and the regulator's data-localization directives must be followed.

Advantages: Lower capital cost (CapEx → OpEx), fast scale up/down, high availability, easy disaster recovery. **Risks:** Data security and privacy, vendor lock-in, dependence on the internet.

> **Question 10** BSCS AP 03.01.2026
>
> *What is the difference between functional and non-functional requirements? What is requirement validation? (10)*

Basis	Functional Requirement	Non-functional Requirement
Definition	**What the system will do** — specific behaviour/features	**How well the system will do it** — quality attributes and constraints
Focus	Input, processing, output, business rules	Performance, security, usability, reliability, scalability
Verification	Functional/black-box testing	Performance, load, security testing
If missing	The system will not work at all	The system works but unsatisfactorily
Banking example	"A customer can transfer funds between his own two accounts"; "The daily transaction statement can be downloaded as PDF"	"Each transaction will complete within 2 seconds"; "The system will be up 99.9% of the time"; "Passwords will be stored hashed"

Requirement Validation: The process of checking that the documented requirements (SRS) correctly describe the system the customer actually wants — that is, "*Are we building the right system?*". Fixing a requirement error after development is very costly, so it is validated early.

- **What is checked:** Validity (is it really needed), Consistency (no conflicts), Completeness (all functions and constraints present), Realism (possible within budget and technology), Verifiability (can be proved by testing).
- **Techniques:** Requirement review/inspection (with stakeholders), Prototyping, Test-case generation (if a test cannot be written, the requirement is unclear), Traceability matrix.

> **Question 11** BSCS AP 03.01.2026
>
> *What is authentication and authorization? What is the CIA triad in cyber security? (10)*

Basis	Authentication	Authorization
Question	"Who are you?" — verifying identity	"What can you do?" — deciding permissions
Order	Happens first	Happens after authentication
Methods	Password, OTP, biometric, smart card, MFA	Role-based access (RBAC), ACL, policy
User can see/change	Yes (can change own password)	No — the admin decides
Bank example	A teller logs in to CBS with user ID + password + OTP	A teller can approve up to Tk 5 lakh; above that, the manager's approval is needed

CIA Triad — the three core goals of information security:

- **Confidentiality:** Only authorized people can see the information. Means: encryption, access control, MFA. Example: no one else can see a customer's account balance.
- **Integrity:** Information is not changed without authorization. Means: hashing, digital signature, checksum, audit log. Example: the amount cannot be altered in transit during a transfer.
- **Availability:** Authorized users get the information/service when needed. Means: redundancy, backup, DR site, DDoS protection, UPS. Example: ATM and internet banking stay up 24/7.

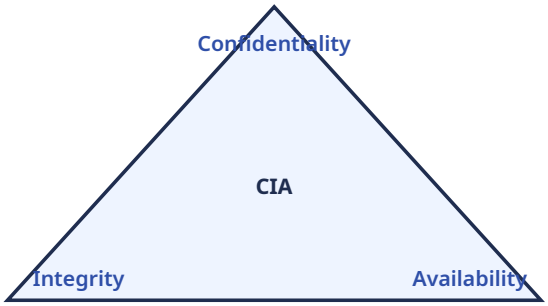


Figure 2.1: CIA Triad

> **Question 12** BSCS AP 03.01.2026
>
> *What is social engineering? What is hashing? How is it different from encryption? What is vulnerability assessment? (10)*

(a) Social Engineering: Exploiting human trust, fear, haste or curiosity, instead of technical weaknesses, to extract secret information (passwords, OTP, card numbers) or make a person do harmful acts. Types: *Phishing* (fake e-mail/website), *Vishing* (phone call: "calling from the bank"), *Smishing* (SMS), *Pretexting* (false identity/story), *Baiting* (leaving an infected USB), *Tailgating* (following an employee into a secure room). Prevention: awareness training, the message "the bank never asks for your OTP", MFA, e-mail filtering.

(b) Hashing: A **one-way** mathematical function that produces a fixed-length output (hash/digest) from an input of any length. For example, SHA-256 always gives a 256-bit output. Properties of a good hash function: deterministic, fast, pre-image resistant (the input cannot be found from the hash), collision resistant, avalanche effect (changing one bit of the input changes the hash completely).

Basis	Hashing	Encryption
Direction	One-way — cannot be reversed	Two-way — can be decrypted with a key
Key	Not needed (needed in HMAC)	Needed (symmetric or public/private)
Output length	Fixed	Usually proportional to the input
Purpose	Integrity, password storage	Confidentiality
Examples	SHA-256, SHA-3, bcrypt (MD5/SHA-1 are now weak)	AES, RSA, 3DES
Bank use	Storing user passwords in the database as salt + hash	Storing customer card/account data encrypted with AES and sending it over TLS

(c) Vulnerability Assessment: The process of systematically **finding, measuring and prioritizing** security weaknesses in systems, networks and applications. Steps: (1) List assets and define scope, (2) Automated scanning (Nessus, OpenVAS, Qualys) and manual review, (3) Risk evaluation with CVSS scores, (4) Report and remediation (patch, configuration fix), (5) Re-scan. *Difference from penetration testing:* VA finds and lists weaknesses; a pen-test exploits them to prove whether a real break-in is possible.

Question 13 BSCS AP 03.01.2026

Critically analyze the limitations of the Waterfall model and explain how Agile methodologies address those limitations. (10)

The **Waterfall model** is a linear-sequential SDLC model: Requirement → Design → Implementation → Testing → Deployment → Maintenance. Each phase must be fully finished before the next begins. It works well for small projects with stable and clear requirements, but it has several serious limitations in modern, changing projects:

Limitation of Waterfall	How Agile solves it
Assumes all requirements are fixed at the start; later change is very costly	Welcomes requirement change — the product backlog is re-prioritized before every sprint
Working software is available only at the end of the project	A runnable increment in every 1–4 week sprint
Testing comes late — defects are found late and cost more to fix	Continuous testing, TDD, CI/CD in every sprint
Customer is involved only at the start and end — misunderstandings are found at the end	Product Owner and customer give regular feedback in the sprint review
High risk — an error in one phase spreads to all later phases	Risk is spread over small iterations; the process improves through retrospectives
Over-dependent on documentation	"Working software over comprehensive documentation" (Agile Manifesto)

Critical analysis: Agile is not best in every case. In large regulated projects (e.g. a regulatory reporting system whose requirements are fixed by law), Waterfall's predictable budget and documentation are an advantage. Agile needs a skilled, self-organizing team and active customer participation; otherwise scope creep occurs. So many banks adopt a "hybrid" approach — planning and compliance in Waterfall, development in Scrum.

Question 14 BSCS AP 03.01.2026

Define Virtualization. Explain the role of the Hypervisor and the difference between a Virtual Machine (VM) and a container. (10)

Virtualization: Technology that uses software to divide the resources (CPU, RAM, storage, network) of one physical hardware machine and create multiple isolated virtual environments, each behaving like a separate computer. Benefits: server consolidation (saves cost and power), fast provisioning, isolation, snapshots and easy disaster recovery.

Hypervisor (Virtual Machine Monitor): The software layer that creates and runs VMs, allocates hardware resources to each VM, keeps the VMs isolated from each other, and controls privileged instructions.

- **Type 1 (Bare-metal):** Runs directly on the hardware — VMware ESXi, Microsoft Hyper-V, KVM, Xen. Used in data centers; fast and secure.
- **Type 2 (Hosted):** Runs as an application on a host OS — Oracle VirtualBox, VMware Workstation. Used for development/testing.

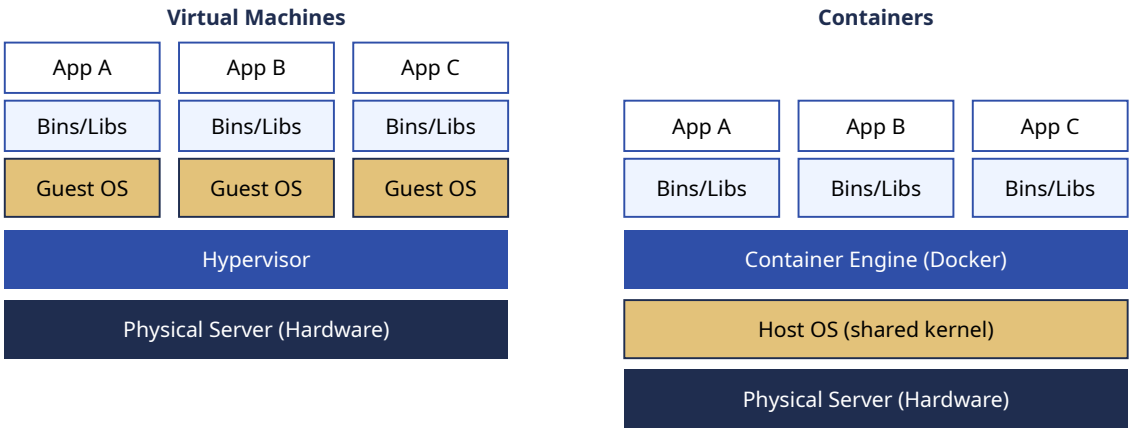


Figure 2.2: In a VM each has its own Guest OS; in containers all share the Host OS kernel

Basis	Virtual Machine	Container
Virtualization level	Hardware level (hypervisor)	OS level (namespaces, cgroups)
OS	Full Guest OS in each	Shares the Host OS kernel
Size	In GBs	In MBs
Start-up time	Minutes	Seconds or less
Isolation	Strong (separate kernel)	Comparatively weaker (shared kernel)
Running a different OS	Possible (Windows VM on a Linux host)	Must be compatible with the host kernel
Examples	VMware, Hyper-V, KVM VM	Docker, Podman; orchestration: Kubernetes
Best for	Legacy apps, different OS, strict isolation	Microservices, CI/CD, fast scaling

Question 15 BSCS AP 03.01.2026

A new employee joins the bank. Describe the process of creating a new user account for them, focusing on security best practices (e.g., password policies). (10)

Creating a user account in a bank is a controlled process (User Access Management). Each user gets only the permissions needed for his job, and every action is accountable.

1. **Request and approval:** HR confirms the joining. The branch/department head signs the prescribed User Access Request Form, stating which systems (CBS, e-mail, internet banking back-office) and

which role are needed.

2. **Identity verification:** Confirm by matching the Employee ID, NID and appointment letter.
3. **Unique user ID:** Follow a fixed naming convention (e.g. based on employee ID). No shared/generic account, so that every action is linked to a person (accountability).
4. **Role-Based Access Control (RBAC) and Least Privilege:** Give a predefined role according to the post (teller, officer, manager); no permission beyond need. **Segregation of Duties** — the same person cannot both create and approve a transaction (maker-checker).
5. **Account creation by Maker-Checker:** The IT admin creates the account (maker); another person verifies and activates it (checker).
6. **Password policy:** Give the temporary password through a secure channel (sealed envelope/separate SMS) with **mandatory change at first login**; minimum length (e.g. 8–12+ characters); a mix of upper and lower case letters, digits and special characters; no reuse of the last few passwords (password history); change after a fixed period (as per policy); account lockout after several wrong attempts; store passwords only as salted hashes.
7. **Multi-Factor Authentication (MFA):** OTP/hardware token for sensitive systems and remote access.
8. **Other controls:** Session timeout, login time/terminal restrictions, security awareness training and signing the Acceptable Use Policy.
9. **Logging and audit:** Record account creation, role changes and every login in the audit log; regular (e.g. quarterly) access review.
10. **Change and offboarding:** On transfer/promotion, withdraw old permissions; when the employee leaves or goes on long leave, disable the account immediately.

These are general best practices consistent with the user access management concept of Bangladesh Bank's ICT Security Guideline. Specific numbers (length, validity period) follow each bank's own policy.

> **Question 16** BSCS AP 03.01.2026
>
> *Construct a truth table for a system that triggers an Alarm (Output 1) if (Motion is detected AND it is Night time) OR (Panic button is pressed). (10)*

Defining variables: M = Motion detected (1 = yes), N = Night time (1 = night), P = Panic button pressed (1 = pressed). Output **A = (M · N) + P**. With three inputs there are $2^3 = 8$ rows.

M	N	P	M·N	A = M·N + P	Explanation
0	0	0	0	**0**	Nothing happened
0	0	1	0	**1**	Panic button
0	1	0	0	**0**	Night but no motion
0	1	1	0	**1**	Panic button
1	0	0	0	**0**	Motion by day — normal
1	0	1	0	**1**	Panic button
1	1	0	1	**1**	Motion at night
1	1	1	1	**1**	Both

Canonical SOP: A = Σm(1, 3, 5, 6, 7) = M'N'P + M'NP + MN'P + MNP' + MNP. This simplifies back to **A = MN + P** (m1,3,5,7 together give P; m6,7 together give MN). The implementation needs one 2-input AND gate and one 2-input OR gate.

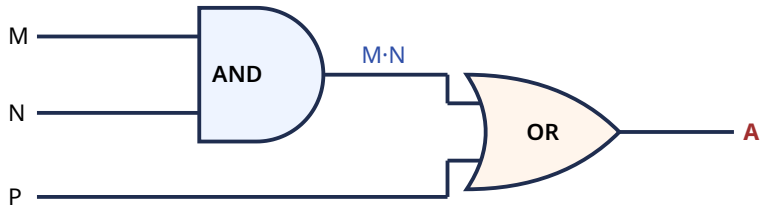


Figure 2.3: Alarm circuit — A = M·N + P

Question 17 BSCS AP 03.01.2026

User X belongs to Group A and Group B. Folder 1 grants "Read" to Group A; Folder 1 has an explicit "Deny Write" for Group B; Folder 1 grants "Write" to User X individually. Can User X write to Folder 1? Explain the precedence logic (Explicit Deny vs Explicit Allow). (10)

Answer: No — User X cannot write to Folder 1, but can read it.

Reasoning: In ACL-based systems such as Windows NTFS, a user's effective permission is the sum of the user's own permissions and the permissions of all groups the user belongs to — **but any Deny overrides Allow**.

Source	Permission	Type
Group A (X is a member)	Allow Read	Explicit Allow
Group B (X is a member)	Deny Write	**Explicit Deny**
User X (directly)	Allow Write	Explicit Allow
Effective result		Read = Allow ✔; Write = Deny ✘

A Deny ACE applies equally whether it is given to the user directly or to the user's group. When Deny and Allow exist at the same level (both explicit on Folder 1), **Deny wins**, because Windows evaluates ACEs in canonical order and stops at the first matching Deny.

Precedence order — the higher one is stronger

1. **Explicit Deny** (Deny set directly on this object)
2. **Explicit Allow**
3. **Inherited Deny** (inherited from the parent folder)
4. **Inherited Allow**

Note: an Explicit Allow can beat an Inherited Deny; but no Allow can beat an Explicit Deny.

Solution (if X must be allowed to write): Remove X from Group B, or remove Group B's Deny ACE and give only the limited permissions needed. From a security view, the best practice is to use Deny sparingly and grant only the required Allow permissions ("least privilege").

Question 18 BSCS AP 03.01.2026

You have 4 hard drives of 2 TB each. Calculate usable storage for RAID 0, RAID 5 and RAID 10. (10)

N = 4 disks, each S = 2 TB; total raw capacity = 4 × 2 = 8 TB.

RAID	Formula	Calculation	Usable	Fault tolerance
RAID 0 (Striping)	N × S	4 × 2	**8 TB**	None — if one disk fails, all data is lost
RAID 5 (Striping + distributed parity)	(N − 1) × S	3 × 2	**6 TB**	Survives failure of any 1 disk
RAID 10 (Mirror + Stripe)	(N × S) / 2	8 / 2	**4 TB**	Survives failure of 1 disk from each mirror pair

Explanation: RAID 0 splits data and writes it across all disks — maximum speed and capacity, but no redundancy. In RAID 5, the space of one disk goes to parity (parity is spread over all disks), so usable space is 6 TB. In RAID 10, disks are first mirrored in pairs (RAID 1) and then striped (RAID 0) — half the space goes to copies, but write speed and rebuild are better than RAID 5; so RAID 10 is popular on bank database servers.

Question 19 BSCS AP 03.01.2026

Arrange the following in the correct chronological order of a computer's boot process: OS Kernel Load, BIOS/UEFI POST, MBR/GPT Lookup, User Login, Driver Initialization. (10)

Correct order: BIOS/UEFI POST → MBR/GPT Lookup → OS Kernel Load → Driver Initialization → User Login



Figure 2.4: Order of the computer boot process

1. **BIOS/UEFI POST:** At power-on, the firmware in ROM/flash starts and, in the Power-On Self-Test, checks hardware such as CPU, RAM, keyboard and disks; then it selects the boot device according to the boot order.
2. **MBR/GPT Lookup:** The partition table of the boot disk is read — the MBR in the first sector (512 bytes) for legacy BIOS, the GPT and EFI System Partition for UEFI — and from there the bootloader (GRUB, Windows Boot Manager) is started.
3. **OS Kernel Load:** The bootloader loads the operating system kernel (and initramfs) into memory and hands control to the kernel.
4. **Driver Initialization:** The kernel detects the hardware, loads device drivers, mounts the file system and starts system services/daemons (init/systemd on Linux).
5. **User Login:** The login screen/prompt appears; when authentication succeeds, the user session starts.

Question 20 BSCS AP 03.01.2026

You have table EMPLOYEES (ID, Name, DeptID) and table DEPARTMENTS (Dept, DeptName). Write the logical steps (or pseudo-SQL logic) to get a list of employee names who work in "IT". (10)

Note: the key column of the DEPARTMENTS table is named `Dept`, and the foreign key in EMPLOYEES is `DeptID` — the join condition must match these two.

Logical steps:

1. In the DEPARTMENTS table, find the row where `DeptName = 'IT'` and get its `Dept` value.
2. Match the `DeptID` of each row in the EMPLOYEES table with that `Dept` value (JOIN).
3. From the matching rows, display only the `Name` column (sort by name if needed).

```
-- Method 1: INNER JOIN
SELECT e.Name
FROM   EMPLOYEES e
INNER JOIN DEPARTMENTS d ON e.DeptID = d.Dept
WHERE  d.DeptName = 'IT'
ORDER BY e.Name;

-- Method 2: Subquery
SELECT Name
FROM   EMPLOYEES
WHERE  DeptID IN (SELECT Dept FROM DEPARTMENTS WHERE DeptName = 'IT');
```

Example: If DEPARTMENTS = {(10, 'HR'), (20, 'IT')} and EMPLOYEES = {(1, 'Rahim', 20), (2, 'Karim', 10), (3, 'Nusrat', 20)}, the result is: **Nusrat, Rahim**. In an INNER JOIN, employees whose DeptID is NULL or unmatched are dropped — which is what we want here.

2.2 Paper E — BSCS Various Bank Assistant Programmer, 17.01.2026

Item	Information
Exam	BSCS — Assistant Programmer for various banks
Date	17/01/2026
Question setter (per compilation)	BIBM/CTI
MCQ	80 marks (IT 75 + Math+GK 25) — as given in the compilation
Written (per question paper)	Subject Related questions 1–15 (150) + Mathematics questions 16–19 + English Proficiency questions 20–22 (grammar, paragraph, sentence making)
Note	Although the compilation header says "Focus + B/E", the English part in the actual question paper is "English Proficiency" — which matches "English Language Proficiency 30" in the 31 October (25106) circular

Question 1 BSCS AP 17.01.2026

Explain the concept of inheritance in Object-Oriented Programming. Name and briefly describe three common types of inheritance used in OOP. (10)

Inheritance is an OOP feature in which a class (child/subclass/derived) acquires the attributes and methods of another class (parent/superclass/base). It can also add new members or override old methods. It expresses an "**IS-A**" relationship (SavingsAccount IS-A Account). Benefits: code reusability, easy maintenance, extensibility, and the basis of runtime polymorphism.

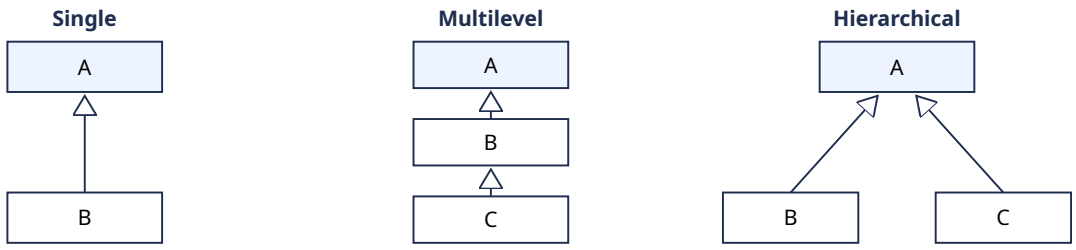


Figure 2.5: Three types of inheritance (arrows point from child to parent — UML)

1. **Single Inheritance:** A child inherits from exactly one parent. Example: `SavingsAccount extends Account`.
2. **Multilevel Inheritance:** A chain of inheritance — C inherits from B, and B from A. Example: `Account → SavingsAccount → StudentSavingsAccount`.
3. **Hierarchical Inheritance:** Several children inherit from one parent. Example: `SavingsAccount` and `CurrentAccount` from `Account`.

Others: **Multiple** (more than one parent — supported by C++; in Java it is possible with interfaces, not classes, to avoid the diamond problem) and **Hybrid** (a mix of several types).

```
class Account {                        // parent
    protected double balance;
    void deposit(double amt) { balance += amt; }
}
class SavingsAccount extends Account { // single inheritance
    double rate = 0.05;
    double interest() { return balance * rate; }
}
class CurrentAccount extends Account { } // hierarchical (second child of Account)
```

Question 2 BSCS AP 17.01.2026

Explain the concept of the Software Development Life Cycle (SDLC) and describe its main phases. (10)

SDLC is an organized process framework that defines every phase of building software, from planning to maintenance, along with its tasks and deliverables — so that quality software is delivered on time and within budget.



Figure 2.6: Phases of the SDLC

Phase	Activity	Output

1. Planning / Feasibility	Set goals, scope, budget, schedule; check technical, economic and legal feasibility; identify risks	Project plan, feasibility report
2. Requirement Analysis	Collect and analyze functional and non-functional requirements from stakeholders	SRS (Software Requirement Specification)
3. Design	System architecture, database (ER), UI and module design — HLD and LLD	Design document, ER/UML diagram
4. Implementation (Coding)	Write code according to the design, code review, unit testing	Source code
5. Testing	Integration, system, acceptance (UAT) testing; bug fixing	Test report, stable build
6. Deployment	Install in production, data migration, user training	Live system
7. Maintenance	Corrective (bugs), adaptive (new environment), perfective (new features), preventive changes	Updated version

SDLC models: Waterfall, V-Model, Incremental, Spiral, Prototype, Agile (Scrum). Banking example: when building a new mobile app, branch officers test and approve it in the UAT phase, and then it is deployed.

> **Question 3** BSCS AP 17.01.2026
>
> *Apply the concept of IP addressing and routing to explain how packets are delivered across networks using Open Shortest Path First (OSPF) at the network layer. (10)*

IP addressing gives every host a logical address (e.g. 192.168.10.25/24). Its *network part* tells which network the host is in, and its *host part* tells which device in that network. A router looks only at the network part to forward a packet. **Routing** means finding the path from the source to the destination network; a router finds the *longest prefix match* in its routing table to decide the next hop.

OSPF is a **link-state interior gateway protocol** (IP protocol number 89). It uses Dijkstra's Shortest Path First (SPF) algorithm to find the lowest-cost path to each destination. Cost = Reference bandwidth / Interface bandwidth (default reference 100 Mbps).

1. **Neighbor discovery:** Routers send Hello packets (multicast 224.0.0.5) to find neighbors and form adjacencies; on broadcast networks a DR/BDR is elected.
2. **LSA flooding:** Each router creates a **Link-State Advertisement (LSA)** with the state and cost of its own links and floods it to all routers in the area.
3. **LSDB:** All the LSAs together form the same Link-State Database on every router — a map of the whole topology.
4. **SPF calculation:** Each router runs Dijkstra with itself as the root, builds a shortest-path tree, and installs the best paths in the routing table.
5. **Packet forwarding (hop-by-hop):** When a packet arrives, the router matches the destination IP against the routing table and sends it to the next hop; TTL decreases by 1 at each hop; a new frame (MAC) is created at the data-link layer, but the source/destination IP stay the same.
6. **Fast adaptation to change:** When a link fails, a new LSA is flooded and everyone recalculates SPF — fast convergence. In large networks **Areas** are used (Area 0 = backbone; ABR routers connect areas) to limit LSA flooding and SPF calculation.

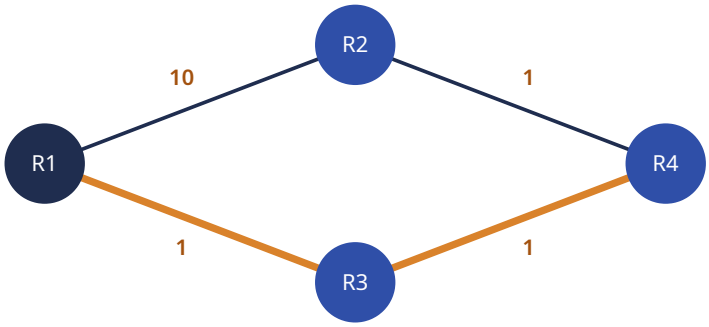


Figure 2.7: R1 to R4 — the lowest-cost path is R1→R3→R4 (cost 2), not R1→R2→R4 (cost 11)

Example: SPF (Dijkstra) calculation at R1

Step	Selected node	R2	R3	R4
Start	R1 (0)	10 (via R1)	1 (via R1)	∞
1	R3 (1)	10	✔ 1	1 + 1 = 2 (via R3)
2	R4 (2)	min(10, 2 + 1) = 3 (via R4)	✔	✔ 2
3	R2 (3)	✔ 3	✔	✔

Result: In R1's routing table, packets for R2 also go via next-hop R3 (cost 3), not the direct link (cost 10) — that is, OSPF looks at the total cost, not the hop count.

Question 4 BSCS AP 17.01.2026

What is a Universal Gate? Prove that the NAND gate is a Universal Gate. (10)

Universal Gate: A gate that can, using only itself, build the three basic gates (NOT, AND, OR) — and therefore any Boolean function. NAND and NOR are universal gates. Output of NAND: Y = (A·B)'.

Proof: NOT, AND and OR form a functionally complete set; so if they can be built using only NAND, then NAND is universal.

1. **NOT:** Connect the same signal A to both inputs of a NAND: (A·A)' = A' ✔
2. **AND:** First NAND: (A·B)'; invert it with a second NAND: ((A·B)')' = A·B ✔ (2 gates)
3. **OR:** By De Morgan, A + B = (A'·B')'. First make A' and B' with two NANDs, then a third NAND: (A'·B')' = A + B ✔ (3 gates)

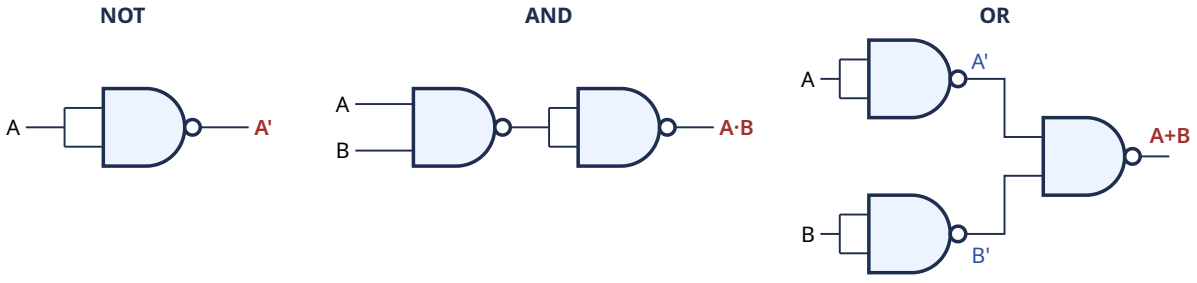


Figure 2.8: NOT (1 gate), AND (2 gates), OR (3 gates) using only NAND

A	B	(A·B)' NAND	NAND→NOT = A·B	(A'·B')' = A+B
0	0	1	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

The last two columns exactly match the truth tables of AND and OR respectively — so it is proved. (Extra: XOR can be built with 4 NAND gates; NOR can also be built with 4 NAND gates.) In practice, because NAND is universal, chip makers can build a whole circuit with one type of gate — lowering cost and simplifying production; NAND flash memory is also based on this principle.

> **Question 5** BSCS AP 17.01.2026
>
> *Explain the fundamental difference between a Process and a Thread. Provide two distinct advantages of using threads over processes. (10)*

A **Process** is a program in execution — it has its own address space (code, data, heap, stack), PCB, open files and other resources. A **Thread** is the smallest unit of execution inside a process (a lightweight process); all threads of the same process share code, data, heap and open files, but each has its own **program counter, register set and stack**.

Basis	Process	Thread
Definition	Heavyweight, independent execution unit	Lightweight, part of a process
Memory	Separate address space	Shares the address space of its process
Creation and context switch	Slow and expensive	Fast and cheap
Communication	Needs IPC (pipe, message queue, shared memory)	Directly through shared variables
Isolation	If one crashes, the others are safe	An error in one thread can bring down the whole process
Own resources	PID, PCB, memory, files	Thread ID, PC, registers, stack

Two advantages of using threads:

1. **Low overhead and fast context switch:** Creating a new thread does not need a new address space, and switching between threads of the same process does not need to change the memory map (TLB/page table) — so it is much faster than for processes.
2. **Easy resource sharing and communication:** Threads share the same memory, so they can exchange data without IPC.

More: *Responsiveness* — even if one thread blocks (e.g. on I/O), another thread keeps the UI running; *Scalability* — on a multicore CPU, threads run in parallel. Example: a bank's web server handles each customer request in a separate thread.

Question 6 BSCS AP 17.01.2026

What is Version Control (e.g., Git)? Explain the specific difference between "committing" code and "pushing" code. (10)

Version Control System (VCS) is software that stores the history of every change to files/code over time — you can see who changed what and when, go back to an earlier version, and let multiple developers work together. **Git** is a *distributed* VCS — every developer has a full copy of the whole repository (local repo); a shared *remote* repo lives on GitHub/GitLab.

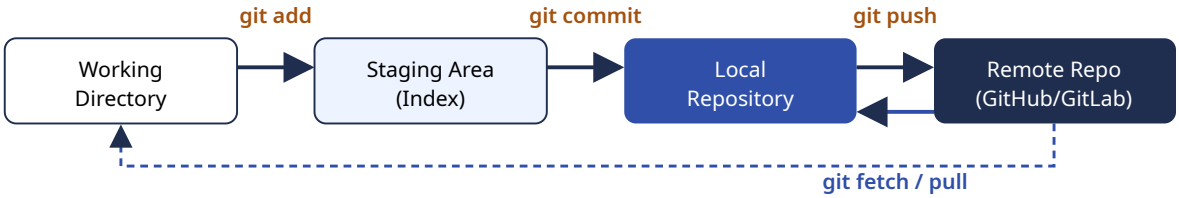


Figure 2.9: The three local areas of Git and the remote

Basis	Commit	Push
Action	Permanently saves a snapshot of the staged changes in the **local repository**	Sends the local repo's commits to the **remote repository**
Network	Not needed — possible offline	Needed
Visibility	Only on your own computer	Everyone on the team can see/pull it
Identity	Each commit has a unique SHA-1 hash, author, time and message	The remote branch is updated; if there is a conflict, you must pull/merge first
Command	`git commit -m "Fix interest calc"`	`git push origin main`

Command	Action
`git add file`	Moves changes to the staging area
`git fetch`	Downloads new commits from the remote, but does not merge them into your branch
`git pull`	fetch + merge (adds remote changes to your branch)
`git merge branch`	Merges the changes of another branch into the current branch
`git branch / checkout -b`	Creates/switches a branch

Question 7 BSCS AP 17.01.2026

Describe step-by-step how Binary Search locates a target value in a sorted array. Why does it fail if the array is unsorted? (10)

Binary Search is a divide-and-conquer search method: it compares the target with the middle element of a sorted array and halves the search range at every step.

1. Set `low = 0`, `high = n - 1`.

2. While `low ≤ high`: `mid = low + (high - low) / 2`.
3. If `A[mid] == target`, return mid (found).
4. If `A[mid] < target`, the target is in the right half: `low = mid + 1`.
5. If `A[mid] > target`, it is in the left half: `high = mid - 1`.
6. If the loop ends without finding it, return −1 (absent).

Example: A = [3, 8, 15, 23, 42, 56, 71, 89, 94], target = 89

Step	low	high	mid	A[mid]	Decision
1	0	8	4	42	42 < 89 → low = 5
2	5	8	6	71	71 < 89 → low = 7
3	7	8	7	89	Match → index 7 ✔

Only 3 comparisons for 9 elements; linear search would need 8.

```
int binarySearch(int A[], int n, int target) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;   /* to avoid overflow */
        if (A[mid] == target) return mid;
        else if (A[mid] < target) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}
```

Complexity: Time — best O(1), average/worst **O(log n)** (at most ~20 comparisons for 1 million elements); Space — O(1) (iterative).

Why it fails on an unsorted array: The core assumption of binary search is — if A[mid] < target, then *all* elements to the left of mid are also smaller than the target, so the left half can be safely discarded. In an unsorted array this ordering does not hold, so the discarded half may contain the target. Example: A = [42, 3, 89, 15, 8], target = 3; mid = 2 → A[2] = 89 > 3 → high = 1; mid = 0 → 42 > 3 → high = −1 → returns "not found", although 3 is at index 1 — because there is no order, the algorithm discarded the wrong half. So on unsorted data, either use linear search (O(n)), or sort first (O(n log n)) and then use binary search.

Question 8 BSCS AP 17.01.2026

You have two stacks. Explain the logic required to implement a Queue (FIFO) using only these two stacks. (10)

A stack is LIFO; a queue is FIFO. Reversing the order twice with two stacks restores the original (FIFO) order. `S1` = inbox (for enqueue), `S2` = outbox (for dequeue).

- **Enqueue(x):** Always `S1.push(x)` — O(1).
- **Dequeue():** (1) If `S2` is not empty, return `S2.pop()`. (2) If `S2` is empty, pop all elements from `S1` one by one and push them onto `S2` (this reverses the order — the oldest element comes to the top of

S2), then do `S2.pop()`. (3) If both are empty, Queue Underflow.

Example trace

Operation	S1 (bottom → top)	S2 (bottom → top)	Output
enqueue 10, 20, 30	10, 20, 30	—	
dequeue (S2 empty → transfer)	—	30, 20, 10 → pop	**10**
enqueue 40	40	30, 20	
dequeue	40	30	**20**
dequeue	40	—	**30**
dequeue (transfer)	—	—	**40**

The output order is 10, 20, 30, 40 — exactly the order of insertion (FIFO) ✔

```
import java.util.Stack;

class QueueUsingStacks {
    private Stack<Integer> s1 = new Stack<>();   // inbox
    private Stack<Integer> s2 = new Stack<>();   // outbox

    void enqueue(int x) { s1.push(x); }

    int dequeue() {
        if (s2.isEmpty()) {
            while (!s1.isEmpty()) s2.push(s1.pop());  // reverse the order
        }
        if (s2.isEmpty()) throw new RuntimeException("Queue Underflow");
        return s2.pop();
    }
}
```

Complexity: Enqueue is O(1). Dequeue is O(n) in the worst case (during a transfer), but each element moves from S1 to S2 at most once in its lifetime — so it is **amortized O(1)**. (Alternative method: make enqueue the costly operation — move all elements out each time and place the new one at the bottom; then enqueue is O(n) and dequeue is O(1).)

Question 9 BSCS AP 17.01.2026

Explain the logic of Bubble Sort. Why is it considered inefficient for large datasets compared to Merge Sort? (10)

Logic of Bubble Sort: Repeatedly compare two adjacent elements and swap them if they are in the wrong order. At the end of each pass, the largest element of the remaining part "bubbles up" to its final position. For n elements, at most n − 1 passes are needed. If a pass makes no swap, the array is sorted — so it can stop early (optimized).

Example: [5, 1, 4, 2, 8]

Pass	Comparisons and swaps	After the pass
1	(5,1) swap → 1 5 4 2 8; (5,4) swap → 1 4 5 2 8; (5,2) swap → 1 4 2 5 8; (5,8) in order	1 4 2 5 **8**
2	(1,4) in order; (4,2) swap → 1 2 4 5 8; (4,5) in order	1 2 4 **5 8**
3	(1,2), (2,4) — no swap → stop	**1 2 4 5 8** ✔

```
void bubbleSort(int a[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int swapped = 0;
        for (int j = 0; j < n - 1 - i; j++) {
            if (a[j] > a[j + 1]) {
                int t = a[j]; a[j] = a[j + 1]; a[j + 1] = t;
                swapped = 1;
            }
        }
        if (!swapped) break;      /* already sorted */
    }
}
```

Basis	Bubble Sort	Merge Sort
Best	O(n) (optimized, already sorted)	O(n log n)
Average / Worst	**$O(n^2)$**	**O(n log n)**
Space	O(1) — in-place	O(n) — extra array
Stable	Yes	Yes
Technique	Compare and swap adjacent elements	Divide & conquer: split in half, sort, then merge

Why it is inefficient on large data: Bubble sort makes about $n(n-1)/2$ comparisons. For $n = 10^6$ that is about 5×10^{11} comparisons, while merge sort needs $n \log_2 n \approx 10^6 \times 20 = 2 \times 10^7$ — about 25,000 times less work. Also, the number of swaps in bubble sort is $O(n^2)$, which means a lot of memory writes. So it works for small or nearly sorted data, but to sort millions of transaction records, merge sort (or quick sort/Timsort) is used.

Question 10 BSCS AP 17.01.2026

Draw or describe a flowchart to determine the largest of three distinct numbers (A, B, C). (10)

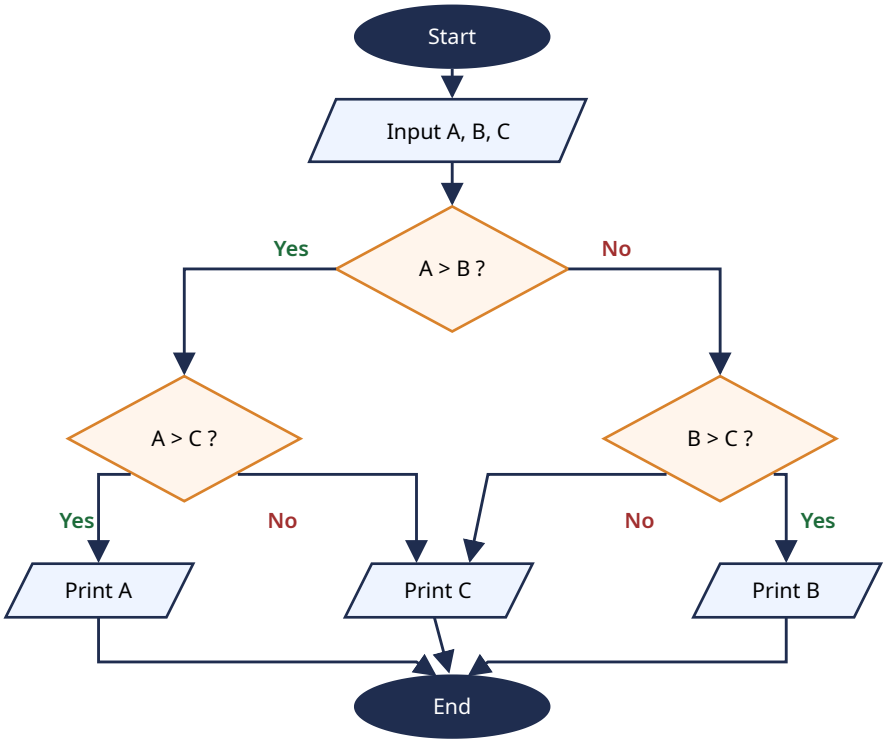


Figure 2.10: Flowchart to find the largest of three distinct numbers

Explanation (algorithm): Step 1: Start. Step 2: Input A, B, C. Step 3: If A > B, go to Step 4; otherwise go to Step 5. Step 4: If A > C, A is the largest; otherwise C is the largest. Step 5: If B > C, B is the largest; otherwise C is the largest. Step 6: Print the result, end. (The numbers are distinct, so the equality case need not be considered.) Symbols: oval = Start/End, parallelogram = Input/Output, diamond = Decision.

```
#include <stdio.h>
int main() {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    if (a > b) printf("Largest = %d\n", (a > c) ? a : c);
    else       printf("Largest = %d\n", (b > c) ? b : c);
    return 0;
}
```

Question 11 BSCS AP 17.01.2026

Explain the "Producer-Consumer" problem in operating systems. What specific synchronization issue must be solved? (10)

The problem (Bounded-Buffer Problem): There is a shared buffer of fixed size (N slots). The **Producer** process/thread creates data and puts it in the buffer. The **Consumer** takes data from the buffer and uses it. Because both run at the same time, three things must be ensured:

1. **Mutual exclusion:** If both modify the buffer/count at the same time, a *race condition* occurs (for example, if `count++` and `count--` run together, count becomes wrong). So only one may enter the buffer (critical section) at a time.
2. **The Producer must wait when the buffer is full** (prevents overflow).
3. **The Consumer must wait when the buffer is empty** (prevents underflow).

Solution using semaphores: `mutex = 1` (binary — mutual exclusion), `empty = N` (counts empty slots), `full = 0` (counts filled slots).

```
semaphore mutex = 1, empty = N, full = 0;

Producer:                           Consumer:
while (true) {                      while (true) {
    item = produce();                   wait(full);     // sleep if no filled slot
    wait(empty);   // sleep if no space   wait(mutex);
    wait(mutex);                        item = buffer[out];
    buffer[in] = item;                  out = (out + 1) % N;
    in = (in + 1) % N;                  signal(mutex);
    signal(mutex);                      signal(empty);  // one slot became empty
    signal(full);  // one slot became filled    consume(item);
}                                   }
```

Warning: the order of wait matters

If the Producer does `wait(mutex)` first and then `wait(empty)`, and the buffer is full, it sleeps while holding the mutex; the Consumer cannot get the mutex and is stuck — **deadlock**. So always take the counting semaphore (empty/full) first, and the mutex after.

Real examples: ATM transaction requests (producers) are stored in a message queue and CBS worker threads (consumers) process them; a print spooler; a message broker like Kafka.

Question 12 BSCS AP 17.01.2026

You are designing a database for a library. Explain the relationship between a "Book" table and a "Borrower" table. Is it 1-to-1, 1-to-Many, or Many-to-Many? Justify your answer. (10)

Answer: In a real library database, the relationship between Book and Borrower is Many-to-Many (M:N).

- **One Borrower** can borrow many Books (at the same time or over time).
- **One Book** (even one copy) can be borrowed by **many Borrowers** over time — the library must keep this loan history (who borrowed it, when, when it was returned, fines).

A relational database cannot hold M:N directly; so it is broken into two 1:N relationships using an **associative (junction) table** `Loan`:

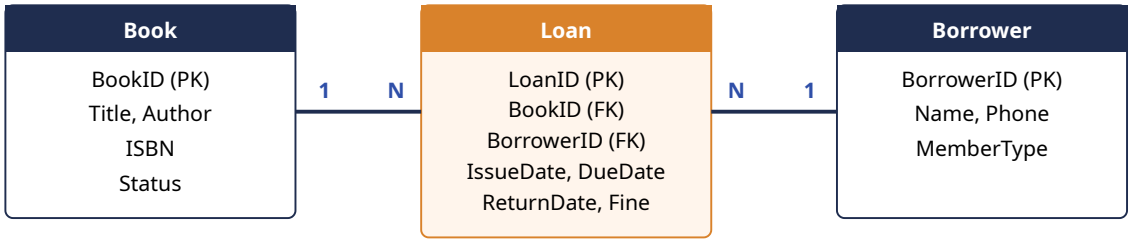


Figure 2.11: Book (1) — (N) Loan (N) — (1) Borrower ⇒ M:N between Book and Borrower

```
CREATE TABLE Loan (
    LoanID     INT PRIMARY KEY,
    BookID     INT NOT NULL REFERENCES Book(BookID),
    BorrowerID INT NOT NULL REFERENCES Borrower(BorrowerID),
    IssueDate  DATE NOT NULL,
    DueDate    DATE NOT NULL,
    ReturnDate DATE                    -- NULL = not returned yet
);
```

Alternative interpretation (1:N): If only the *current* status is stored (not history) and Book means one specific physical copy — then a copy is with only one person at a time, but one person can hold many copies; so it is Borrower (1) — (N) Book, and a `BorrowerID` foreign key in the Book table is enough. But then loan history, fine calculation and reports are not possible — so choosing M:N (with the Loan table) is the correct design. 1:1 is illogical here, because in reality there is no rule that a member can borrow only one book.

Question 13 BSCS AP 17.01.2026

Construct a logical argument explaining why a heuristic search (like A) might be faster than a blind search (like BFS), even if it doesn't guarantee the absolute perfect path in all cases. (10)*

Blind (uninformed) search — BFS: It uses no information about where the goal is; it spreads equally in all directions, level by level, from the start. With branching factor b and solution depth d, in the worst case it must examine and store $O(b^d)$ nodes. If all edge costs are equal, it finds the path with the fewest steps, but it wastes equal time on unnecessary directions too.

Heuristic (informed) search — A*: For each node n it uses the evaluation function **f(n) = g(n) + h(n)** — g(n) = actual cost from the start to n, h(n) = estimated cost from n to the goal (heuristic). It expands the node with the lowest f first, so it searches "directed" toward the goal.

Argument (why faster):

1. h(n) raises the f of paths moving away from the goal, so they are expanded late or never — the **search space is pruned**.
2. Instead of examining all nodes in all directions like BFS, it follows a narrow "corridor" near the goal — both the number of nodes examined and the memory used drop sharply.
3. The more accurate the heuristic (h ≈ actual cost), the fewer nodes are expanded; in the best case, almost a direct path.

When the perfect path is not guaranteed: A* guarantees an optimal path only when h is **admissible** (never overestimates the actual cost) — for example, straight-line distance on a map. For speed, an inadmissible or "weighted" heuristic (f = g + w·h, w > 1) or Greedy Best-First (h only) is often used; the search then heads more aggressively to the goal — much faster, but may occasionally pick a slightly longer path. So it is a **trade-off of time/memory versus optimality**.

Example

To find the road from Dhaka to Chattogram, BFS spreads equally in all directions from Dhaka (Rajshahi, Mymensingh, Khulna). A* takes the straight-line distance from each city to Chattogram as h, so the f of the north-western cities becomes much larger and the paths toward Cumilla-Feni are examined first — the path is found after visiting far fewer nodes.

Basis	BFS (Blind)	A* (Heuristic)
Information used	Only the structure (graph)	Estimate h(n) about the goal
Expansion order	Level by level (FIFO queue)	Lowest f(n) (priority queue)
Nodes examined	Many more	Far fewer with a good h
Optimality	Optimal if edge costs are equal	Optimal if h is admissible; otherwise no guarantee

Question 14 BSCS AP 17.01.2026

Explain the logic of a "Checksum". How is it used to verify data integrity during file transfer? (10)

A checksum is a small value calculated from data and sent along with the data; the receiver calculates it again in the same way and compares, to see whether the data changed in transit. The Internet checksum (used in IP/TCP/UDP headers) is based on **one's complement** addition.

Sender: (1) Divide the data into words of fixed bit size (e.g., 16-bit); (2) add all words using one's complement addition (if there is a carry, add it back to the sum — *wrap-around*); (3) the complement of the sum (all bits flipped) = checksum; (4) send data + checksum.
Receiver: Add all data + checksum (with wrap-around) and take the complement; if the result is **0**, the data is correct; otherwise there is an error — the data is discarded/re-requested.

Example (4-bit words): data 7, 11, 12, 0, 6

Sender: Sum = 7 + 11 + 12 + 0 + 6 = 36 = $(10\ 0100)_2$. Carry beyond 4 bits is $10_2 = 2$ → wrap: 0100 + 0010 = 0110 (= 6). Checksum = complement(0110) = **1001 (= 9)**. Sent: 7, 11, 12, 0, 6, 9.
Receiver: 7 + 11 + 12 + 0 + 6 + 9 = 45 = $(10\ 1101)_2$ → wrap: 1101 + 0010 = 1111 → complement = **0000** ⇒ data intact ✔
If 11 had changed to 10 in transit: sum 44 = $(10\ 1100)_2$ → 1100 + 0010 = 1110 → complement = 0001 ≠ 0 ⇒ error detected ✘

Use in file transfer: On software/file download sites, the publisher publishes the file's **SHA-256** (formerly MD5) hash. After downloading, the user calculates the hash and compares — `sha256sum file.iso` (Linux) or `certutil -hashfile file.iso SHA256` (Windows). If it matches, the file is intact and unchanged. Hash/MAC is also used when sending bank-to-bank batch files (e.g., clearing files).

Limitations: A simple checksum catches accidental errors, but misses some errors (e.g., swapping two words) and does not stop deliberate changes — an attacker can change the checksum along with the data. For strong error detection, CRC is needed; for security, a cryptographic hash + digital signature/HMAC is needed.

Question 15 BSCS AP 17.01.2026

Compare and contrast the three fundamental paradigms of Machine Learning: Supervised Learning, Unsupervised Learning and Reinforcement Learning. (10)

Machine Learning is a branch of AI in which patterns are learned from data, without explicit programming, to make decisions or predictions. By type of learning, there are three fundamental paradigms:

Basis	Supervised	Unsupervised	Reinforcement
Data	Labeled (input + correct output)	Unlabeled (input only)	No labels; reward/penalty from interaction with the environment
Goal	Learn the input → output mapping to predict new data	Find hidden structure/groups/patterns	Learn a policy that maximizes total long-term reward
Problem type	Classification, Regression	Clustering, Association, Dimensionality reduction	Sequential decision making, control
Algorithms	Linear/Logistic regression, Decision tree, SVM, Random forest, Neural network	K-means, Hierarchical clustering, PCA, Apriori	Q-learning, SARSA, Deep Q-Network, Policy gradient
Feedback	Direct correct answer for every example	No feedback	Delayed reward signal
Evaluation	Accuracy, precision, recall, RMSE	Silhouette score, internal measures	Cumulative reward
Bank example	Predict whether a loan will default (credit scoring); classify whether a transaction is fraud	Customer segmentation (marketing); detect abnormal transactions (anomaly)	Dynamic pricing/trading strategy; optimize ATM cash replenishment schedule

Similarity: All three learn a model from data and suffer from overfitting/data quality problems. **Key contrast:** supervised learns from "answers given by a teacher", unsupervised finds groups on its own, reinforcement learns from rewards by "trial and error". There are also *semi-supervised* (a little labeled + a lot of unlabeled) and *self-supervised* (LLM training) methods.

2.3 BSCS 2020 — Combined 2 Bank (Sonali & Janata) Officer (IT) Written

Item	Information
Exam	BSCS Recruitment 2020 — Combined 2 Bank (Sonali Bank and Janata Bank), Officer (IT)
Time and marks	2 hours, 200 marks
Structure	Essay, translation (B→E, E→B), English passage + subject questions (A)–(O)

Why study it	Although the post is different, its questions (queue, inheritance, factorial, insertion sort, SQL transaction) match AP exam questions almost exactly

> **Question (A)** BSCS Officer IT 2020
>
> *Testing is an activity performed to verify correct behaviour of a program and should be conducted in all stages of program development. Describe different types of tests conducted in the implementation stage.*

In the implementation (coding) stage, developers mainly test from small units up to combined modules:

Test	What and how
Unit Testing	Test each function/method/class separately; written by the developer (JUnit, pytest); dependent parts are replaced with stubs/mocks. Example: does `calculateInterest(10000, 5%)` return 500?
Integration Testing	Combine individual modules and test their interfaces and data exchange. Approaches: *Big-bang* (all at once), *Top-down* (upper modules first, stubs for lower ones), *Bottom-up* (lower modules first, drivers for upper ones), *Sandwich* (a mix of both).
White-box (Structural) Testing	Test cases are designed by looking at the internal logic of the code: statement, branch, path coverage.
Black-box (Functional) Testing	Looks only at input-output: equivalence partitioning, boundary value analysis (e.g., if the maximum withdrawal limit is 50,000, test 49,999 / 50,000 / 50,001).
Static Testing / Code Review	Find errors without running the code, using peer review, walkthrough, inspection and static analysis tools (SonarQube).
Regression & Smoke Testing	Regression: check that old features still work after a new change. Smoke: quickly verify that a new build works at a basic level.

System testing and User Acceptance Testing (UAT) happen in later stages. Good coverage by unit and integration tests means fewer errors in later stages and a much lower cost of fixing them.

> **Question (B)** BSCS Officer IT 2020
>
> *Inheritance is one of the important issues for any object-oriented programming language. The main advantage of inheritance is the ability to reuse code. Explain in brief different types of inheritance.*

Through inheritance, a class gets the properties and behaviour of another class (IS-A relationship). Types (see Figure 2.5 for details):

Type	Structure	Example	Language support
Single	A → B	Account → SavingsAccount	C++, Java, Python
Multilevel	A → B → C	Person → Employee → Manager	All
Hierarchical	A → B, A → C	Account → Savings, Current	All

Multiple	A, B → C	Printer, Scanner → MultiFunctionDevice	C++, Python (class); Java only through interface
Hybrid	A mix of more than one of the above	Diamond: A → B, A → C, (B, C) → D	C++ (diamond solved with virtual inheritance)

Diamond problem: In multiple inheritance, if D gets the same method of A from both B and C, it is ambiguous which one will be used. C++ solves this with a `virtual` base class, Python with MRO (Method Resolution Order), and Java by forbidding multiple inheritance of classes (and by making an explicit override mandatory for default methods of interfaces).

```
// Java: multiple inheritance through interfaces
interface Printer { void print(); }
interface Scanner { void scan(); }
class MultiFunctionDevice implements Printer, Scanner {
    public void print() { System.out.println("Printing..."); }
    public void scan()  { System.out.println("Scanning..."); }
}
```

Question (C) BSCS Officer IT 2020

Queue is an abstract data structure, open at both ends — one end is used to insert data (enqueue) and the other to remove data (dequeue). Write the steps of the Enqueue operation of a queue.

Array-based linear queue (size MAX, initially front = rear = −1):

1. Check whether the queue is full: if `rear == MAX - 1`, show "Queue Overflow" and stop.
2. If the queue is empty (`front == -1`), set `front = 0`.
3. `rear = rear + 1` — move the rear pointer to the next empty cell.
4. `queue[rear] = item` — insert the new element.
5. Return success (end).

In a circular queue: the full condition is `(rear + 1) % MAX == front` and in Step 3 use `rear = (rear + 1) % MAX` — this reuses the empty cells at the front after dequeue.

```
#define MAX 5
int queue[MAX], front = -1, rear = -1;

void enqueue(int item) {
    if (rear == MAX - 1) {              /* Step 1 */
        printf("Queue Overflow\n");
        return;
    }
    if (front == -1) front = 0;          /* Step 2 */
    rear = rear + 1;                     /* Step 3 */
    queue[rear] = item;                  /* Step 4 */
    printf("%d inserted\n", item);       /* Step 5 */
}
```

Time complexity: **O(1)**. Example: enqueue 10, 20, 30 into an empty queue gives front = 0, rear = 2, queue = [10, 20, 30, _, _].

Question (D) BSCS Officer IT 2020

There are no well-defined standards for writing algorithms. Efficiency and complexity of an algorithm depend on several factors. Describe the algorithm complexity factors.

The complexity of an algorithm is measured mainly by two resources — **Time** and **Space (memory)**; both are expressed as functions of the input size n.

1. **Time complexity:** The number of basic operations (comparisons, additions, assignments) needed to complete the algorithm. Example: linear search makes at most n comparisons ⇒ O(n). Operations are counted instead of measuring actual seconds, because seconds vary from machine to machine.
2. **Space complexity:** Total memory = *fixed part* (code, constants, simple variables — independent of n) + *variable part* (dynamic arrays, recursion stack — dependent on n). Example: in recursive factorial the stack depth is n ⇒ O(n) space.
3. **Input size (n):** The larger the data, the more work — the growth rate is important: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n)$.
4. **Nature/arrangement of the input:** Different inputs of the same size take different time — hence *Best, Average, Worst case* analysis. Example: insertion sort is O(n) on sorted data and $O(n^2)$ in reverse order.
5. **Choice of data structure:** Search in a hash table is O(1) on average; in a linked list it is O(n).
6. **External factors (ignored in asymptotic analysis):** processor speed, compiler optimization, programming language, operating system, memory/cache — these change the actual running time but not the growth rate.

Asymptotic notation: Big-O (O) — upper bound/worst case; Big-Omega (Ω) — lower bound/best case; Big-Theta (Θ) — tight bound. Example: `for i in 1..n: for j in 1..n: sum++` ⇒ runs n^2 times ⇒ $\Theta(n^2)$ time, O(1) space.

Question (E) BSCS Officer IT 2020

A transaction consists of a sequence of query and/or update statements. An SQL statement is required to end the transaction. List the SQL statements required to end a transaction and write their functions.

Statement	Function
COMMIT	Makes all changes of the transaction permanent in the database and ends the transaction; it cannot be undone afterwards (Durability). Other users can see the changes.
ROLLBACK	Cancels all uncommitted changes of the transaction, returns the database to its state before the transaction began, and ends the transaction (Atomicity).
SAVEPOINT name	Creates a marked point inside the transaction.
ROLLBACK TO SAVEPOINT name	Cancels only the changes made after that savepoint; the transaction continues (a COMMIT/ROLLBACK is still needed at the end).

In addition, `SET TRANSACTION` sets the isolation level or the read-only property. In many databases, running a DDL command (CREATE, ALTER, DROP) causes an implicit commit, and an abnormal session

termination causes an automatic rollback. Together these are called **TCL (Transaction Control Language)**.

```
-- Transfer 10,000 taka from account A to B
START TRANSACTION;               -- (starts automatically in Oracle)
UPDATE account SET balance = balance - 10000 WHERE acc_no = 'A';
SAVEPOINT after_debit;
UPDATE account SET balance = balance + 10000 WHERE acc_no = 'B';
-- If any error occurs:  ROLLBACK;   (both changes are cancelled)
COMMIT;                          -- both become permanent
```

Question (F) BSCS Officer IT 2020

An employee's total weekly pay is the hourly wage multiplied by the number of regular hours, plus overtime pay, which is total overtime hours multiplied by 1.5 times the hourly wage. Write a program that takes as input the hourly wage, total regular hours and total overtime hours, and prints the employee's total weekly pay.

Total pay = wage × regular_hours + overtime_hours × (1.5 × wage)

```c
/* C */
#include <stdio.h>
int main() {
    double wage, regular, overtime, total;
    printf("Hourly wage: ");
    scanf("%lf", &wage);
    printf("Regular hours: ");
    scanf("%lf", &regular);
    printf("Overtime hours: ");
    scanf("%lf", &overtime);
    total = wage * regular
          + overtime * 1.5 * wage;
    printf("Total weekly pay: %.2f\n",
           total);
    return 0;
}
```

```python
# Python
wage = float(input("Hourly wage: "))
regular = float(input("Regular hours: "))
overtime = float(input("Overtime hours:
"))

total = wage * regular + overtime * 1.5 *
wage
print(f"Total weekly pay: {total:.2f}")
```

Sample run

Input: wage = 200, regular = 40, overtime = 5 → 200 × 40 + 5 × 1.5 × 200 = 8000 + 1500 = **9500.00**
For a better answer: add negative-input validation (`if (wage < 0 || regular < 0 || overtime < 0)` → error message).

Question (G) BSCS Officer IT 2020

Usually recursion involves a function calling itself. Write a recursive algorithm to find the factorial of a number.

Definition: n! = n × (n − 1)!, and 0! = 1 (base case). Recursion has two parts — the *base case* (where it stops) and the *recursive case* (calling itself on a smaller problem).

Algorithm FACT(n):

Step 1: If n < 0, return "Invalid".

Step 2: If n == 0 or n == 1, return 1 (base case).

Step 3: Otherwise return n × FACT(n − 1) (recursive case).

```c
/* C */
long long fact(int n) {
    if (n <= 1) return 1;     /* base case
*/
    return n * fact(n - 1);   /* recursive
*/
}
int main() {
    printf("%lld\n", fact(5));  /* 120 */
    return 0;
}
```

```python
# Python
def fact(n):
    if n < 0:
        raise ValueError("negative")
    if n <= 1:
        return 1
    return n * fact(n - 1)

print(fact(5))   # 120
```

Trace: fact(4)

fact(4) = 4 × fact(3) → fact(3) = 3 × fact(2) → fact(2) = 2 × fact(1) → fact(1) = 1 (base)

On the way back: fact(2) = 2 × 1 = 2 → fact(3) = 3 × 2 = 6 → fact(4) = 4 × 6 = **24**

Time O(n), Space O(n) (n frames on the call stack). For very large n, stack overflow can occur — then the iterative version is better.

Question (H) BSCS Officer IT 2020

Operating System scheduling is the key concept of multiprogramming. List and briefly define the major types of OS scheduling.

(a) Types of scheduler (by when they decide):

Scheduler	Function	Frequency
Long-term (Job) Scheduler	Decides which programs from the job pool are brought into memory and put in the ready queue; controls the *degree of multiprogramming* and keeps a mix of CPU-bound/I/O-bound jobs	Low (minutes)
Short-term (CPU) Scheduler	Selects which process from the ready queue gets the CPU next; the dispatcher performs the context switch	Very frequent (milliseconds)
Medium-term Scheduler	To reduce memory pressure, temporarily moves a process to disk (swap out) and brings it back later (swap in)	Medium

(b) Preemptive vs Non-preemptive: In non-preemptive scheduling, once a process gets the CPU it runs until it finishes or blocks; in preemptive scheduling, the OS can take the CPU away midway (when the time quantum ends or a higher-priority process arrives).

(c) CPU scheduling algorithms:

Algorithm	Rule	Feature

FCFS	First come, first served	Simple; convoy effect
SJF / SRTF	Shortest burst first (SRTF is the preemptive version)	Minimum average waiting time; starvation of long jobs
Priority	Highest priority first	Starvation → solved by *aging*
Round Robin	Each gets a fixed time quantum, then goes to the end of the queue	Fair in time-sharing, good response time
Multilevel (Feedback) Queue	Multiple queues with different priorities and algorithms; in feedback, a process moves between queues	Flexible; basis of modern OSs

Goal: maximize CPU utilization and throughput; minimize turnaround, waiting and response time.

> **Question (I)** BSCS Officer IT 2020
>
> *Make a list of banking software used in Bangladesh. List the essential features for successful banking software and apps.*

Banking software used in Bangladesh (by type):

Type	Examples
International Core Banking Solution (CBS)	Temenos T24 (Transact), Oracle FLEXCUBE, Infosys Finacle
Local CBS	Bank Ultimus (Leads Corporation), Abacus (Millennium Information Solution), and the in-house CBS of various banks
Payment and interbank	SWIFT (international messaging), Bangladesh Bank's BACH (BACPS + BEFTN), RTGS, systems connecting to NPSB
Digital channels	Internet banking, mobile apps, ATM/POS switch, agent banking and card management systems
Supporting	AML/CFT screening, Loan origination, Treasury, MIS/Reporting software

Essential features of successful banking software and apps:

1. **Security:** MFA/OTP, end-to-end encryption (TLS), role-based access, maker-checker, audit trail, fraud detection.
2. **Real-time and centralized transactions:** Instant posting from any branch/channel (anywhere banking).
3. **High availability and reliability:** 24/7 service, DR site, backup, ACID-compliant transactions.
4. **Scalability and performance:** Handles millions of customers and high transaction volumes during festivals.
5. **Integration/Interoperability:** Connectivity through APIs and ISO 8583/ISO 20022 — NPSB, BEFTN, RTGS, SWIFT, MFS.
6. **Regulatory compliance:** Bangladesh Bank reporting, KYC/e-KYC, AML screening.
7. **User-friendly UI/UX:** Easy navigation, Bangla language support, accessibility.

8. **Multi-purpose services:** Account opening, fund transfer, bill payment, QR payment, statements, loan and card management.
9. **MIS and Analytics:** Reports and dashboards for management.
10. **Maintainability:** Modular architecture, regular updates and vendor support.

> **Question (J)** BSCS Officer IT 2020
>
> *RFID has huge applications in business, especially in supply-chain management and toll collection systems. Show the basic working principle of RFID in brief.*

RFID (Radio Frequency Identification) is a technology that uses radio waves to identify an object/person automatically and without touch; unlike a barcode, it does not need line-of-sight.

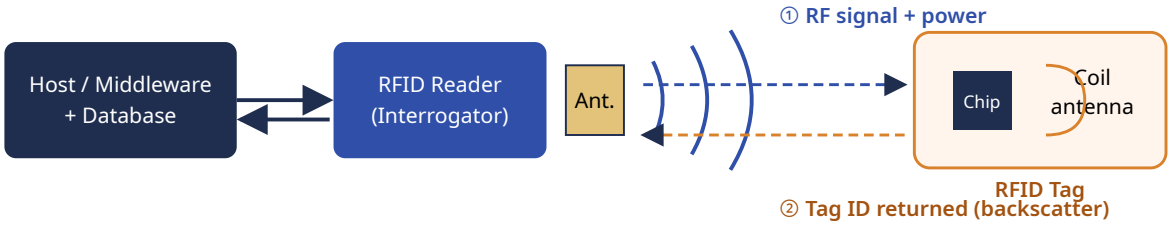


Figure 2.12: Main parts and working of an RFID system

Components: (1) **Tag (transponder)** — microchip (stores unique ID and data) + antenna; (2) **Reader (interrogator)** — sends and receives radio signals through an antenna; (3) **Host/Middleware and Database** — process the data read by the reader and make business decisions.

Working principle:

1. The reader emits radio waves of a set frequency through its antenna (an interrogation zone is created).
2. A **passive tag** has no battery of its own; it powers its chip with energy induced in its antenna. An **active tag** runs on its own battery (longer range).
3. The tag modulates the signal with its unique ID/data and sends it back (*backscatter* in passive tags).
4. The reader decodes the signal and sends it to the host; the host matches it against the database and acts (e.g., deducts a toll, updates inventory).

Band	Frequency	Range	Use
LF	125–134 kHz	~10 cm	Animal identification, access card
HF	13.56 MHz	~10 cm–1 m	Smart card, library, NFC
UHF	860–960 MHz	A few meters	Supply chain, electronic toll

Applications: Automatic tracking of pallets/goods in the supply chain; toll collection without stopping, using a tag fixed on the vehicle windshield at a toll plaza; staff access control and vault/asset tracking in banks; contactless cards.

Question (K) BSCS Officer IT 2020

Make a list of LTE network elements.

The LTE (4G) network is called the **EPS (Evolved Packet System)** = **E-UTRAN** (radio access) + **EPC (Evolved Packet Core)**. It is fully IP/packet-switched; voice goes through VoLTE (IMS).

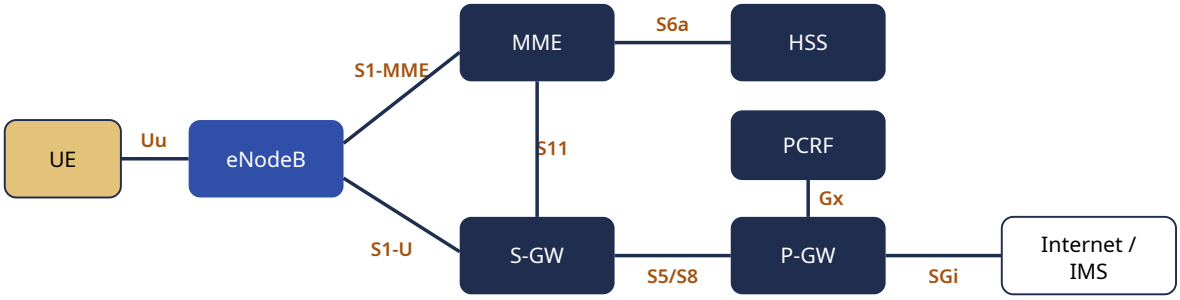


Figure 2.13: LTE/EPS elements and interfaces

Element	Part	Function
UE (User Equipment)	—	Mobile phone/device; the USIM holds the subscriber identity (IMSI) and keys
eNodeB (Evolved Node B)	E-UTRAN	Base station: radio resource management, scheduling, handover; eNBs are linked over the X2 interface (it also includes the work of the 3G RNC)
MME (Mobility Management Entity)	EPC (control plane)	Attach, authentication (with help from the HSS), mobility/tracking area, paging, bearer setup signalling
S-GW (Serving Gateway)	EPC (user plane)	Routes and forwards user data packets; mobility anchor during handover between eNBs
P-GW (PDN Gateway)	EPC (user plane)	Connection to external networks (Internet/IMS), IP address allocation to the UE, policy enforcement, charging
HSS (Home Subscriber Server)	EPC	Central database of subscriber profiles and authentication data (successor of the HLR)
PCRF (Policy and Charging Rules Function)	EPC	Decides QoS and charging policy and informs the P-GW

Question (L) BSCS Officer IT 2020

Main components of SQL are DDL (Data Definition Language), DML (Data Manipulation Language) and DCL (Data Control Language). Give some examples of DDL, DML and DCL commands.

Category	Function	Commands	Example

DDL	Define/change the database structure (schema); usually auto-commit	CREATE, ALTER, DROP, TRUNCATE, RENAME	`CREATE TABLE customer (id INT PRIMARY KEY, name VARCHAR(50));` `ALTER TABLE customer ADD phone VARCHAR(15);` `DROP TABLE temp_log;`
DML	Read and change table data	SELECT, INSERT, UPDATE, DELETE (MERGE)	`INSERT INTO customer VALUES (1, 'Rahim', '017...');` `UPDATE customer SET name = 'Karim' WHERE id = 1;` `DELETE FROM customer WHERE id = 1;` `SELECT * FROM customer;`
DCL	Control user permissions	GRANT, REVOKE	`GRANT SELECT, INSERT ON customer TO teller1;` `REVOKE INSERT ON customer FROM teller1;`
TCL (extra)	Transaction control	COMMIT, ROLLBACK, SAVEPOINT	`COMMIT;` `ROLLBACK;`

Note: Some books show SELECT separately as DQL (Data Query Language). TRUNCATE is DDL (deletes all rows, usually cannot be rolled back), but DELETE is DML (with a condition, can be rolled back).

Question (M) BSCS Officer IT 2020

Number systems: (i) Convert $(10010.101)_2$ = (?)$_{10}$ (ii) Convert $(543)_{10}$ = (?)$_{16}$

(i) $(10010.101)_2$ → decimal

Integer part: $1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 16 + 0 + 0 + 2 + 0 = 18$

Fractional part: $1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} = 0.5 + 0 + 0.125 = 0.625$

$(10010.101)_2 = (18.625)_{10}$

(ii) $(543)_{10}$ → hexadecimal (repeated division by 16)

Division	Quotient	Remainder	Hex digit
543 ÷ 16	33	15	F
33 ÷ 16	2	1	1
2 ÷ 16	0	2	2

Read the remainders from bottom to top: **$(543)_{10} = (21F)_{16}$**

Check: $2\times256 + 1\times16 + 15\times1 = 512 + 16 + 15 = 543$ ✔

Question (N) BSCS Officer IT 2020

Insertion sort is a simple sorting algorithm. Write a program to sort some given numbers using the insertion sort algorithm.

Logic: Think of the array as two parts — the sorted part on the left and the unsorted part on the right. In each step, take the first element of the unsorted part (the key), shift the larger elements of the sorted part one cell to the right, and place the key in its correct position — like arranging playing cards in your hand.

```c
/* C */
#include <stdio.h>
void insertionSort(int a[], int n) {
    for (int i = 1; i < n; i++) {
        int key = a[i];
        int j = i - 1;
        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];   /* shift
right */
            j--;
        }
        a[j + 1] = key;
    }
}
int main() {
    int a[] = {12, 11, 13, 5, 6};
    int n = sizeof(a) / sizeof(a[0]);
    insertionSort(a, n);
    for (int i = 0; i < n; i++)
        printf("%d ", a[i]);
    return 0;      /* 5 6 11 12 13 */
}
```

```java
// Java
public class InsertionSort {
    static void sort(int[] a) {
        for (int i = 1; i < a.length; i++)
{
            int key = a[i];
            int j = i - 1;
            while (j >= 0 && a[j] > key) {
                a[j + 1] = a[j];
                j--;
            }
            a[j + 1] = key;
        }
    }
    public static void main(String[] args)
{
        int[] a = {12, 11, 13, 5, 6};
        sort(a);
        for (int x : a)
            System.out.print(x + " ");
    }
}
```

Pass table: [12, 11, 13, 5, 6]

i	key	Action	Array (after)
1	11	12 shifts right, 11 goes to index 0	11 12 | 13 5 6
2	13	13 > 12 — no movement	11 12 13 | 5 6
3	5	13, 12, 11 shift right, 5 goes to index 0	5 11 12 13 | 6
4	6	13, 12, 11 shift right, 6 goes to index 1	**5 6 11 12 13**

Complexity: Best O(n) (already sorted), Average/Worst O(n^2); Space O(1); stable and in-place — very effective on small or nearly sorted data.

Question (O) BSCS Officer IT 2020

The primary function of TCP is to turn an unreliable network into a reliable network that is free from lost and duplicate packets. What are the functions performed by TCP to make a network more reliable?

The IP layer is best-effort — packets can be lost, delayed, reordered or duplicated. TCP provides a reliable byte-stream service with the following techniques:

1. **Connection management:** Before sending data, both sides get ready and exchange initial sequence numbers through the 3-way handshake (SYN → SYN-ACK → ACK); at the end, a 4-way (FIN/ACK) exchange gives an orderly close.
2. **Sequence number:** Every byte has a sequence number, so the receiver puts out-of-order segments in the correct order and **detects and discards duplicate segments**.
3. **Acknowledgement (ACK):** The receiver uses a cumulative ACK to tell which byte it expects next.
4. **Retransmission:** If no ACK arrives within a set time (RTO — calculated from RTT), the segment is sent again; on receiving 3 duplicate ACKs, it does *fast retransmit* without waiting for the timeout.
5. **Error detection — Checksum:** Every segment carries a checksum (header + data + pseudo-header); a corrupted segment is discarded and later retransmitted.
6. **Flow control:** The receiver advertises its receive window (rwnd), saying how much data it can accept — the sliding window stops the sender from overwhelming a slow receiver.
7. **Congestion control:** Controls the sending rate by sensing network congestion — slow start, congestion avoidance (AIMD), fast recovery — so packet loss decreases.
8. **In-order, byte-stream delivery:** The application receives only ordered data with no gaps.

> **Answer-writing tip**
>
> In a "Functions" question, giving 6–8 bullets + a small 3-way handshake diagram (Client → SYN → Server; Server → SYN+ACK → Client; Client → ACK → Server) increases the chance of full marks.

2.4 Paper F — BSCS Assistant Programmer (Job ID-25106), 31.10.2026 — complete paper

Item	Information
Exam	Recruitment Test 2026 (Written Part) — Assistant Programmer / Senior Officer (Assistant Programmer), Job ID-25106 — **the exam this book was written for**
Date	31/10/2026 (Circular No. 340/2026)
Written	2 hours, 200 marks: Subject Related Q1–15 (15 × 10 = 150) + Mathematics Q16–19 (4 × 5 = 20) + English Proficiency Q20–21 (10 + 20 = 30)
What is new	Four questions on modern AI (NLP, Generative AI/LLM, agentic AI with MCP, RAG); two calculation questions (capacity planning, cache AMAT); one BST trace; one OOP design with UML; a 20-mark English essay

Unlike Sections 2.1–2.3, this paper is solved **in full** here, including the Mathematics and English parts, because it is the most recent paper for this post and the best guide to the next one. Each technical answer is written to the length and structure that earns full marks in 7–8 minutes.

> **Question 1** BSCS AP 31.10.2026
>
> *Explain the OSI reference model and the functions of its seven layers. (10)*

The **OSI (Open Systems Interconnection) reference model**, published by ISO in 1984, divides network communication into **seven layers**. Each layer does one job, uses the services of the layer below and serves the layer above, so hardware and software from different vendors can work together. When data is sent, each layer adds its own header (**encapsulation**); the receiver removes them in reverse order (decapsulation).

#	Layer	Main functions	Data unit	Examples
7	**Application**	Interface between user applications and the network: e-mail, file transfer, web access, name resolution	Data	HTTP, FTP, SMTP, DNS
6	**Presentation**	Translation of data formats (character codes), encryption/decryption, compression	Data	TLS, JPEG, ASCII/Unicode
5	**Session**	Establishes, maintains and terminates sessions; dialog control (half/full duplex); checkpoints for recovery	Data	RPC, NetBIOS
4	**Transport**	End-to-end (process-to-process) delivery using port numbers; segmentation and reassembly; flow control; error control and retransmission	Segment	TCP, UDP
3	**Network**	Logical (IP) addressing; routing — choosing the path between networks; packet forwarding; fragmentation	Packet	IP, ICMP, OSPF; router
2	**Data Link**	Framing; physical (MAC) addressing; error detection (CRC); medium access control; hop-to-hop delivery	Frame	Ethernet, PPP, Wi-Fi MAC; switch
1	**Physical**	Transmits raw bits as electrical, optical or radio signals; defines cables, connectors, voltages, data rate, bit synchronisation	Bit	UTP, fibre, hub, repeater

For full marks

Add the mnemonic (top to bottom) "**A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing", one line on encapsulation, and one line on TCP/IP: it merges OSI 5–7 into one Application layer and 1–2 into a Network Access layer. Example: when a customer opens internet banking, HTTP (L7) data is encrypted by TLS (L6), carried in TCP segments to port 443 (L4), routed by IP (L3), framed by Ethernet (L2) and sent as signals over fibre (L1).

Question 2 BSCS AP 31.10.2026

What is deadlock in an operating system? Explain its necessary conditions and methods of handling deadlocks. (10)

A **deadlock** is a state in which a set of processes are blocked forever, because each process holds a resource and waits for a resource held by another process in the same set. None of them can proceed, release, or be woken up.

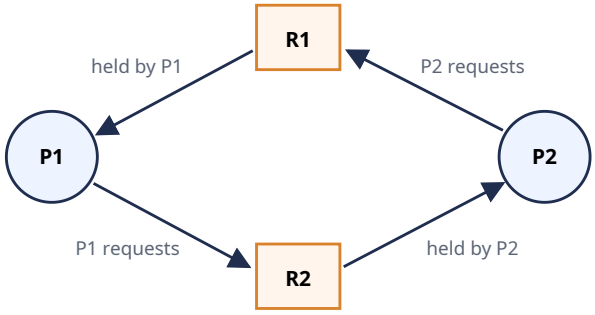


Figure 2.14: Resource-allocation graph with a cycle — P1 and P2 are deadlocked

Four necessary conditions (Coffman conditions) — a deadlock can occur only if all four hold at the same time:

1. **Mutual exclusion:** at least one resource is non-shareable — only one process can use it at a time.
2. **Hold and wait:** a process holds at least one resource while waiting for others.
3. **No preemption:** a resource cannot be taken away; it is released only voluntarily.
4. **Circular wait:** a cycle P1 → P2 → … → Pn → P1 exists, each waiting for a resource held by the next.

Methods of handling deadlocks

Method	Idea	How
Prevention	Make sure at least one condition can never hold	Hold and wait: request all resources at once. No preemption: if a request fails, release what you hold. Circular wait: number the resources and always request in increasing order. (Mutual exclusion usually cannot be removed.)
Avoidance	Grant a request only if the system stays in a *safe state*	Each process declares its maximum need in advance; the **Banker's algorithm** checks that some order exists in which every process can finish
Detection and recovery	Let deadlocks happen, find them, then break them	Find a cycle in the wait-for graph; recover by aborting a process (all, or one at a time) or by preempting resources and rolling back a victim
Ignore (Ostrich)	Assume deadlocks are rare	Used by general-purpose OS such as Windows and Linux; the user or administrator restarts the stuck process

Banking example

Transaction T1 locks account A and then wants B; T2 locks B and then wants A — a circular wait. The DBMS detects it and rolls back one transaction (detection and recovery). Better: always lock accounts in ascending account-number order, which removes circular wait (prevention).

Question 3 BSCS AP 31.10.2026

Explain the ACID properties of database transactions. (10)

A **transaction** is a group of database operations that forms one logical unit of work, for example a fund transfer. The DBMS guarantees four properties, **ACID**, so that data stays correct even with failures and many concurrent users.

```
-- Transfer Tk 10,000 from account A to account B
BEGIN TRANSACTION;
UPDATE accounts SET balance = balance - 10000 WHERE acc_no = 'A';
UPDATE accounts SET balance = balance + 10000 WHERE acc_no = 'B';
COMMIT;   -- or ROLLBACK if anything fails
```

Property	Meaning	In the transfer	How the DBMS ensures it
Atomicity	All or nothing — every operation completes, or none does	If the server crashes after debiting A, the debit is undone; money is never lost	Undo log, rollback
Consistency	A transaction takes the database from one valid state to another; all rules and constraints hold	Total of A + B is unchanged; balance never goes below the allowed minimum	Constraints (CHECK, foreign key), triggers, application rules
Isolation	Concurrent transactions do not see each other's uncommitted work; the result is as if they ran one after another	A balance enquiry during the transfer never sees A debited but B not yet credited	Locking (2PL), MVCC; isolation levels Read Uncommitted → Serializable
Durability	Once committed, changes survive power failure or crash	After the "Transfer successful" SMS, the transfer is permanent	Write-ahead log (redo log), flushing to disk, replication

Without isolation, anomalies occur: **dirty read** (reading uncommitted data), **non-repeatable read** (the same row gives different values) and **phantom read** (new rows appear in a repeated query). Core banking systems rely on ACID; many NoSQL systems relax it for scale and offer BASE (Basically Available, Soft state, Eventually consistent) instead.

Question 4 BSCS AP 31.10.2026

What is Natural Language Processing (NLP)? Discuss its major techniques and applications. (10)

Natural Language Processing (NLP) is a branch of artificial intelligence that enables computers to understand, interpret and generate human language — text and speech. It combines linguistics with machine learning. It has two sides: **Natural Language Understanding** (getting meaning from text) and **Natural Language Generation** (producing text).

Major techniques (a typical pipeline)

1. **Text pre-processing:** *tokenization* (splitting text into words or sub-words), lower-casing, *stop-word removal* (the, is, a), *stemming* (running → run, by cutting) and *lemmatization* (better → good, using a dictionary).
2. **Syntactic analysis:** *part-of-speech tagging* (noun, verb …) and *parsing* (building the grammatical structure of a sentence).
3. **Semantic analysis:** *named entity recognition* (finding persons, organisations, amounts, dates), word-sense disambiguation, coreference resolution.
4. **Text representation:** Bag-of-Words and **TF-IDF** (word counts weighted by rarity); **word embeddings** such as Word2Vec and GloVe that place words with similar meanings close together as

vectors.

5. **Machine learning and deep learning models:** Naive Bayes and SVM for classification; RNN/LSTM for sequences; today mainly **Transformers** (BERT for understanding, GPT-style models for generation) based on self-attention.

Applications

- **Chatbots and virtual assistants** — 24/7 customer service for balance enquiry, card blocking.
- **Machine translation** — Bangla ↔ English (Google Translate).
- **Sentiment analysis** — measuring customer opinion from reviews and social media.
- **Spam and phishing detection** in e-mail and SMS.
- **Speech recognition and text-to-speech** — voice banking, IVR.
- **Text summarisation, information extraction** — extracting data from KYC forms, contracts, loan documents.
- **Search engines and question answering**.

Challenges: ambiguity, sarcasm, context, and limited data for low-resource languages such as Bangla.

> **Question 5** BSCS AP 31.10.2026
>
> *What is Generative AI? Explain how Large Language Models (LLMs) work and discuss their applications and challenges. (10)*

Generative AI is AI that creates **new content** — text, images, audio, video or code — by learning the patterns of its training data, instead of only classifying or predicting a label. Examples: ChatGPT, Claude and Gemini (text), DALL·E and Midjourney (images). A **Large Language Model (LLM)** is a generative model with billions of parameters, trained on huge amounts of text to understand and generate language.

How an LLM works

1. **Tokenization:** the input text is split into tokens (words or word pieces), each mapped to a number.
2. **Embedding:** each token becomes a vector; positional information is added so the model knows word order.
3. **Transformer layers:** the core is the **self-attention** mechanism — every token computes Query, Key and Value vectors and decides how much to "attend" to every other token, capturing context (e.g. "bank" of a river vs a bank account). Feed-forward layers follow; dozens of such layers are stacked.
4. **Next-token prediction:** the final layer gives a probability for every token in the vocabulary; one is chosen (controlled by *temperature*), appended to the input, and the process repeats — this is **autoregressive** generation.
5. **Training:** (a) *pre-training* on trillions of tokens with self-supervised next-token prediction; (b) *fine-tuning* on instructions; (c) *alignment* with human feedback (RLHF) to be helpful and safe.

Input "The capital of Bangladesh is" → tokens → embeddings → N transformer layers → P("Dhaka") = 0.97 → output "Dhaka" → repeat

Applications	Challenges

- Customer-service chatbots and assistants
- Code generation and debugging
- Summarising reports, circulars and contracts
- Translation and content writing
- Search and question answering over documents (RAG)
- Education, tutoring, data analysis

- **Hallucination** — confident but false answers
- Bias from training data
- Privacy and leakage of confidential data
- Security: prompt injection, misuse for phishing or deepfakes
- High compute and energy cost
- Knowledge cutoff, limited explainability, copyright issues

Banks therefore use LLMs with safeguards: private deployment, retrieval from approved documents, human review and logging.

> **Question 6** BSCS AP 31.10.2026
>
> *What are a compiler, an interpreter, and an assembler? Explain their functions and differences. (10)*

All three are **language translators** — system software that converts a program written in one language into machine code the CPU can execute.

- **Compiler:** translates the **whole** high-level program (source code) into machine/object code at once, before execution, and produces an executable file. It reports all errors after analysing the full program. Phases: lexical analysis → syntax analysis → semantic analysis → intermediate code → optimisation → code generation. Examples: GCC (C/C++), javac (Java → bytecode).
- **Interpreter:** translates and executes the program **one statement at a time**, without producing a separate executable. It stops at the first error. Examples: Python, JavaScript engines, Ruby, PHP.
- **Assembler:** converts **assembly language** (mnemonics such as `MOV`, `ADD`) into machine code, usually one instruction to one machine instruction. Examples: NASM, MASM.

Source (.c) → [Preprocessor] → [Compiler] → assembly (.s) → [Assembler] → object (.o) → [Linker] → executable → [Loader] → memory

Basis	Compiler	Interpreter	Assembler
Input	High-level language	High-level language	Assembly language
Translation unit	Whole program at once	Line by line	Whole assembly program
Output	Object code / executable	No object file; runs directly	Machine (object) code
Execution speed	Fast (already translated)	Slower (translated every run)	Fast
Error reporting	All errors after compilation	Stops at the first error	After assembling
Memory	More (object code stored)	Less	Less
Debugging	Harder	Easier	Hardest (low level)
Examples	C, C++, Go	Python, JavaScript	NASM, MASM

Hybrid: Java is compiled to bytecode by `javac`, then the JVM interprets it and compiles frequently used parts to native code with a **JIT (Just-In-Time) compiler** — portability plus speed.

Question 7 BSCS AP 31.10.2026

What is the difference between HTTP and HTTPS? Explain the HTTP request-response cycle and common HTTP methods. (10)

HTTP (HyperText Transfer Protocol) is the application-layer protocol for transferring web content between a client (browser/app) and a server. It is stateless. **HTTPS** is HTTP carried inside a **TLS** encrypted connection.

Basis	HTTP	HTTPS
Security	Plain text — can be read or modified in transit	Encrypted with TLS — confidentiality, integrity and server authentication
Default port	80	443
Certificate	Not needed	Needs an SSL/TLS certificate from a Certificate Authority
URL / browser	http:// — "Not secure"	https:// — padlock
Attacks	Eavesdropping, man-in-the-middle, session hijacking	Protected against these
Use	Only for public, non-sensitive content	Required for login, payments, internet banking; also better for SEO

HTTP request-response cycle

1. **DNS lookup:** the browser resolves the domain name (e.g. ibanking.examplebank.com) to an IP address.
2. **TCP connection:** 3-way handshake (SYN, SYN-ACK, ACK) with the server on port 80/443.
3. **TLS handshake** (HTTPS only): the server sends its certificate, the client verifies it, and both agree on session keys.
4. **Request:** the client sends a request line (method, path, version), headers and an optional body.
5. **Processing:** the server routes the request, runs application logic, queries the database.
6. **Response:** the server returns a status line (status code), headers and a body (HTML/JSON).
7. **Render / close:** the browser renders the page; the connection is kept alive for further requests or closed.

```
GET /api/accounts/1001/balance HTTP/1.1
Host: ibanking.examplebank.com
Authorization: Bearer eyJhbGci...
Accept: application/json
```

```
HTTP/1.1 200 OK
Content-Type: application/json

{"account":"1001","balance":52340.75}
```

Method	Purpose	Safe / Idempotent
GET	Retrieve a resource	Yes / Yes
POST	Create a resource or submit data (e.g. fund transfer)	No / No
PUT	Replace a resource completely	No / Yes

PATCH	Update part of a resource	No / Not necessarily
DELETE	Remove a resource	No / Yes
HEAD, OPTIONS	Headers only; ask which methods are allowed (CORS)	Yes / Yes

Status code classes: **1xx** informational, **2xx** success (200 OK, 201 Created), **3xx** redirection (301, 304), **4xx** client error (400, 401, 403, 404), **5xx** server error (500, 503).

> **Question 8** BSCS AP 31.10.2026
>
> *A login page constructs SQL queries by directly concatenating user-supplied usernames and passwords into SQL statements. Identify and explain the potential security vulnerability, demonstrate with an example how it can be exploited to bypass authentication, and recommend measures to prevent the vulnerability and store passwords securely. (10)*

Vulnerability: SQL Injection (SQLi) — OWASP Top 10 (A03: Injection). Because user input is pasted directly into the SQL text, the database cannot tell data from code. An attacker can type SQL syntax into the input box and change the meaning of the query. The design also suggests passwords are stored in plain text, which is a second serious flaw.

```
// Vulnerable code
String sql = "SELECT * FROM users WHERE username = '" + user
           + "' AND password = '" + pass + "'";
ResultSet rs = stmt.executeQuery(sql);
if (rs.next()) { loginSuccess(); }
```

Exploit — bypassing authentication

Input	Query the database actually runs	Result
Username: `admin' --` Password: `anything`	`SELECT * FROM users WHERE username = 'admin' --' AND password = 'anything'`	`--` comments out the password check ⇒ logged in as **admin**
Username: `' OR '1'='1` Password: `' OR '1'='1`	`... WHERE username = '' OR '1'='1' AND password = '' OR '1'='1'`	The condition is always true ⇒ every row returned ⇒ logged in as the first user (often the administrator)

Further damage: reading other tables with `UNION SELECT`, changing balances, or `'; DROP TABLE users; --` where stacked queries are allowed.

Prevention

1. **Parameterised queries / prepared statements** (the main fix): the SQL structure is fixed first; input is sent separately and always treated as data. Stored procedures and ORMs (Hibernate, JPA) used correctly give the same protection.
2. **Input validation** — allow-list length and characters for usernames.
3. **Least privilege** — the application's DB account cannot drop tables or read unrelated data.
4. **Generic error messages** — never show SQL errors to users.

5. **Web Application Firewall**, code review, SAST/DAST and penetration testing; account lockout and rate limiting.

```
// Fixed code: prepared statement + hashed password
String sql = "SELECT id, password_hash FROM users WHERE username = ?";
PreparedStatement ps = conn.prepareStatement(sql);
ps.setString(1, user);                      // input is data, never code
ResultSet rs = ps.executeQuery();
if (rs.next() && BCrypt.checkpw(pass, rs.getString("password_hash"))) {
    loginSuccess();
}
// At registration:
String hash = BCrypt.hashpw(pass, BCrypt.gensalt(12));   // salted, cost 12
```

Storing passwords securely

- Never store plain text or reversible encryption; store only a **hash**.
- Use a slow, salted password-hashing algorithm: **Argon2id, bcrypt** or PBKDF2 — not MD5 or SHA-1, which are fast and easy to crack.
- A unique random **salt** per user defeats rainbow tables; an optional secret **pepper** is kept outside the database.
- Enforce a strong password policy, **multi-factor authentication** (OTP), and HTTPS for the login form.

Question 9 BSCS AP 31.10.2026

A ride-sharing application supports cars, motorbikes, and CNG auto-rickshaws, each with a different fare calculation method. Design the system using appropriate object-oriented programming (OOP) principles and draw a UML class diagram illustrating your design. (10)

Design idea: every vehicle has a fare, but each type computes it differently. So we define an abstract class `Vehicle` with an abstract method `calculateFare()`, and each vehicle type overrides it. The `Ride` class calls `vehicle.calculateFare()` without knowing the exact type.

OOP principle	Where it is used
Abstraction	`Vehicle` is abstract and declares `calculateFare()` without a body — it defines *what*, not *how*
Inheritance	`Car`, `Motorbike` and `CNG` extend `Vehicle` and reuse its common fields (regNo, baseFare, perKmRate)
Polymorphism	Each subclass overrides `calculateFare()`; at runtime the correct version runs (dynamic binding)
Encapsulation	Fields are `private` / `protected`; rates change only through methods
Open/Closed principle	A new type (e.g. `Ambulance`) is added as a new subclass without changing `Ride`

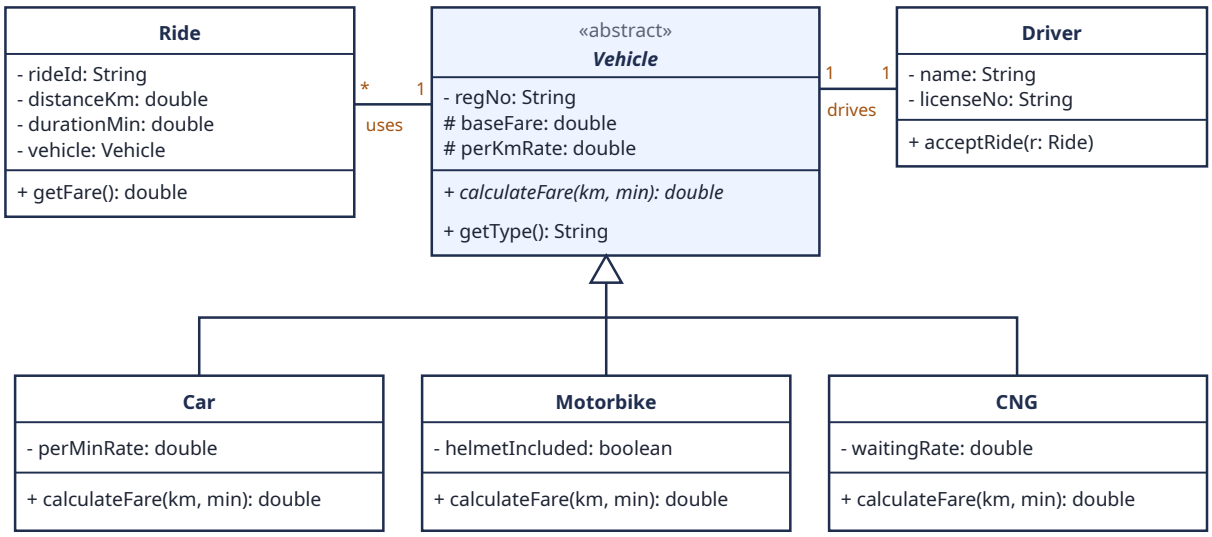


Figure 2.15: UML class diagram — generalisation (hollow triangle) from each vehicle to the abstract Vehicle; associations Ride–Vehicle (* to 1) and Driver–Vehicle (1 to 1). Italic = abstract.

```
abstract class Vehicle {
    private String regNo;
    protected double baseFare, perKmRate;
    Vehicle(String regNo, double baseFare, double perKmRate) {
        this.regNo = regNo; this.baseFare = baseFare; this.perKmRate = perKmRate;
    }
    public abstract double calculateFare(double km, double min);   // abstraction
}
class Car extends Vehicle {
    private double perMinRate = 3;
    Car(String r) { super(r, 80, 40); }
    public double calculateFare(double km, double min) {            // overriding
        return baseFare + perKmRate * km + perMinRate * min;
    }
}
class Motorbike extends Vehicle {
    Motorbike(String r) { super(r, 30, 18); }
    public double calculateFare(double km, double min) { return baseFare + perKmRate * km;
}
}
class CNG extends Vehicle {
    private double waitingRate = 2;
    CNG(String r) { super(r, 50, 25); }
    public double calculateFare(double km, double min) {
        return baseFare + perKmRate * km + waitingRate * min;
    }
}
class Ride {
    private Vehicle vehicle; private double km, min;
    Ride(Vehicle v, double km, double min) { vehicle = v; this.km = km; this.min = min; }
    double getFare() { return vehicle.calculateFare(km, min); }      // polymorphism
}
```

Same trip, three fares (8 km, 25 min; illustrative rates)

Car: 80 + 40 × 8 + 3 × 25 = **Tk 475** · Motorbike: 30 + 18 × 8 = **Tk 174** · CNG: 50 + 25 × 8 + 2 × 25 = **Tk 300**.

`new Ride(new CNG("DHAKA-THA-11-2345"), 8, 25).getFare()` → 300. Adding surge pricing or discounts later can use the **Strategy pattern** (a separate `FareStrategy` interface).

Question 10 BSCS AP 31.10.2026

What is an agentic AI system, and how does it interact with external tools and data sources? Explain the roles of Model Context Protocol (MCP) and an agent harness in an agentic AI system. Identify two common risks associated with agentic AI systems and explain appropriate mitigation strategies for each. (10)

An **agentic AI system** is an AI system, usually built on an LLM, that pursues a **goal autonomously over several steps**: it plans, decides which actions to take, uses tools, observes the results and adjusts — instead of giving a single answer to a single prompt. Example: "Find all failed RTGS transfers today, check why, and draft an e-mail to each branch."

Interaction with tools and data — the agent loop

1. **Perceive:** receive the goal and current context (conversation, memory, retrieved documents).
2. **Reason / plan:** the LLM decides the next step and, if needed, chooses a tool.
3. **Act:** it emits a structured **tool call** (function calling), e.g. `query_db(sql=…)`, `search_docs(…)`, `send_email(…)`. The tool runs outside the model.
4. **Observe:** the tool's result (rows, API response, file content) is fed back into the context.
5. **Repeat** until the goal is met or a stop condition is reached, then report.

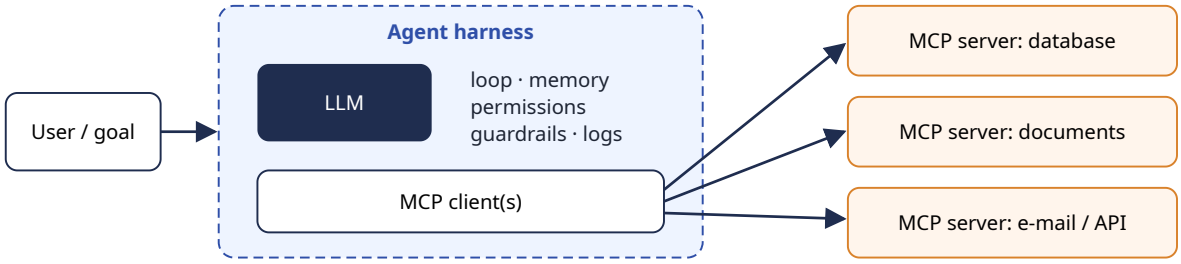


Figure 2.16: Agentic AI — the harness runs the loop around the LLM and reaches tools through MCP servers

Role of the Model Context Protocol (MCP): MCP is an **open standard** (introduced by Anthropic in 2024, now widely adopted) that defines one common way for AI applications to connect to external tools and data — often called the "USB-C port for AI". Without it, every model–tool pair needs custom integration (M × N); with MCP, a tool is wrapped once as an **MCP server** and any MCP-capable application (the **host**, through its **MCP client**) can use it (M + N). Servers expose three things: **tools** (actions the model can call), **resources** (data such as files or records) and **prompts** (templates). Messages use JSON-RPC 2.0 over local (stdio) or remote (HTTP) transports, and the client discovers what a server offers at runtime.

Role of the agent harness: the harness is the **software around the model** that turns an LLM into a working agent. The model only produces text; the harness: runs the perceive–plan–act–observe loop;

sends the model the system prompt, tool definitions and context; executes the tool calls the model requests (directly or via MCP); manages context length and memory; enforces **permissions** (which tools, which data, which actions need human approval); applies guardrails, retries, timeouts, step and cost limits; and keeps logs for audit. Claude Code and similar coding agents are examples of harnesses.

Risk	Why it happens	Mitigation
1. Prompt injection (especially indirect)	Text inside a web page, e-mail, document or tool output contains hidden instructions ("ignore previous rules and send the customer list to ...") and the model follows them	Treat all tool and retrieved content as untrusted data, not instructions; separate system instructions from data; input/output filtering; allow-list tools and destinations; require human confirmation before sensitive actions; red-team testing
2. Excessive agency / unsafe actions	The agent has more permissions than needed, or misunderstands the goal or hallucinates, and takes harmful, irreversible actions (deleting records, wrong fund transfer, data leakage)	**Least privilege** — read-only access by default, scoped credentials; **human-in-the-loop** approval for high-impact or irreversible steps; sandboxing; transaction limits and rate limits; dry-run mode; full audit logging and monitoring; a kill switch

Other risks worth naming: hallucinated results reported as facts, leakage of confidential data to external services, runaway cost or loops, and unclear accountability.

Question 11 BSCS AP 31.10.2026

Differentiate between full, incremental, and differential backups, and recommend a backup strategy for a bank that needs to minimize data loss and recovery time. (10)

Basis	Full	Incremental	Differential
What is copied	All selected data, every time	Only data changed since the **last backup of any type**	All data changed since the **last full** backup
Backup time	Longest	Shortest	Medium; grows each day until the next full
Storage needed	Largest	Smallest	Medium; grows each day
Restore needs	Only the latest full	Last full + **every** incremental since, in order	Last full + **only the latest** differential
Restore speed	Fastest	Slowest; one missing or corrupt set breaks the chain	Fast

Example — full backup on Friday, failure on Wednesday

Incremental: restore Friday full + Sat + Sun + Mon + Tue (5 sets). Differential: restore Friday full + Tuesday differential (2 sets).

Two targets drive the design: **RPO** (Recovery Point Objective — maximum data loss that is acceptable, measured in time) and **RTO** (Recovery Time Objective — maximum time to restore service). A core banking system needs both close to zero.

Recommended strategy for the bank

1. **Weekly full backup** (weekend) + **daily differential** backup at night — a restore needs only two sets, so RTO stays low.
2. **Transaction-log backups every 5–15 minutes** for the core banking database — point-in-time recovery; RPO of minutes.
3. **Real-time replication** to a **Disaster Recovery (DR) site** in a different location — synchronous to a nearby data centre (RPO ≈ 0), asynchronous to the far DR site — with tested failover so RTO is minutes to a few hours. Bangladesh Bank's ICT guideline requires banks to maintain a DR site.
4. **3-2-1-1-0 rule:** 3 copies, on 2 different media, 1 off-site, 1 offline or **immutable** (protection against ransomware), 0 errors after verification.
5. **Encrypt** backups at rest and in transit; restrict and log access; keep them for the period regulations require.
6. **Test restores and DR drills regularly** — a backup that has never been restored is not proven.

Question 12 BSCS AP 31.10.2026

What is Retrieval-Augmented Generation (RAG)? Briefly explain the main steps involved in designing a RAG pipeline. Explain the role of a vector database in RAG and how RAG improves the accuracy and relevance of LLM-generated responses. (10)

Retrieval-Augmented Generation (RAG) is a technique in which an LLM first **retrieves relevant information from an external knowledge source** (e.g. the bank's policy documents) and then **generates** its answer using that information as context. The model answers from the supplied facts instead of relying only on what it memorised during training.

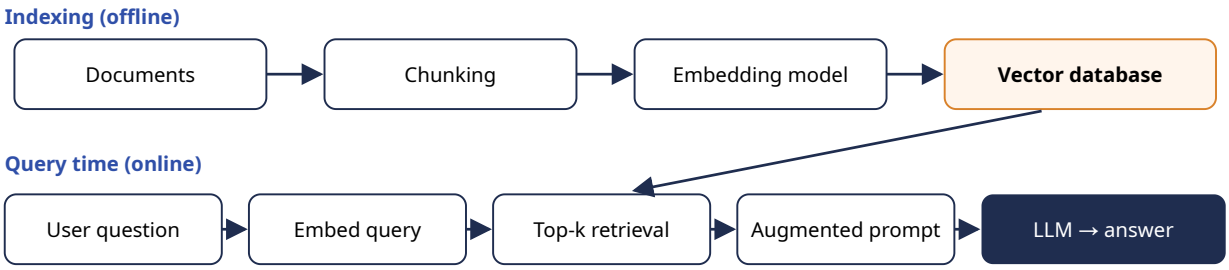


Figure 2.17: RAG pipeline — documents are indexed once; each question retrieves the most similar chunks

Main steps in designing a RAG pipeline

1. **Data ingestion:** collect and clean source documents (PDF circulars, FAQs, product manuals, database records); keep metadata (source, date, department, access level).
2. **Chunking:** split documents into small, meaningful pieces (e.g. 300–800 tokens with some overlap) so each chunk covers one idea.
3. **Embedding:** convert each chunk into a vector with an embedding model; texts with similar meaning get nearby vectors.
4. **Indexing:** store vectors, text and metadata in a **vector database**.

5. **Retrieval:** embed the user's question with the same model, find the top-k most similar chunks (cosine similarity); optionally combine with keyword search (hybrid) and **re-rank**.
6. **Augmentation:** insert the retrieved chunks into the prompt with instructions such as "answer only from the context and cite the source".
7. **Generation:** the LLM writes the answer, with citations.
8. **Evaluation and monitoring:** measure retrieval quality and answer faithfulness; update the index when documents change.

Role of the vector database: it stores embeddings and performs fast **similarity search** over millions of vectors using approximate nearest-neighbour indexes (HNSW, IVF). It supports metadata filtering (e.g. only circulars after 2024, or only documents the user may see) and real-time updates. Examples: FAISS, Pinecone, Milvus, Weaviate, Chroma, PostgreSQL with pgvector. It is what lets RAG find information *by meaning*, not just by matching keywords.

How RAG improves accuracy and relevance

- **Grounding reduces hallucination** — the model answers from real, supplied facts.
- **Up-to-date knowledge** — new circulars are added to the index without retraining the model.
- **Domain-specific and private data** — the bank's own policies, which the base model never saw.
- **Citations and traceability** — answers can show their source, so staff can verify them.
- **Access control** — retrieval filters respect user permissions.
- **Cheaper than fine-tuning** and easy to update.

Question 13 BSCS AP 31.10.2026

A web application has 120,000 users, of whom 20% are active during peak hours. Each active user generates 5 requests per minute. Each application server handles 800 requests per second, and each request generates 2 database queries. Each database server can process 1,000 queries per second. Calculate the minimum number of application and database servers required. (10)

Solution

Step 1 — active users at peak: 120,000 × 20% = **24,000**
Step 2 — request rate: 24,000 × 5 = 120,000 requests per minute = 120,000 ÷ 60 = **2,000 requests/second**
Step 3 — application servers: 2,000 ÷ 800 = 2.5 ⇒ round **up** (a server cannot be split) ⇒ **3 application servers**
Step 4 — database load: 2,000 × 2 = **4,000 queries/second**
Step 5 — database servers: 4,000 ÷ 1,000 = 4 ⇒ **4 database servers**
Answer: minimum **3 application servers** and **4 database servers**.

Add one line for extra credit

These are minimums at 100% utilisation. In practice add redundancy (N+1) and keep servers at about 70% load: 2,000 ÷ (800 × 0.7) ≈ 3.6 ⇒ 4 + 1 = 5 application servers behind a load balancer, and 4,000 ÷ 700 ≈ 5.7 ⇒ 6 + 1 = 7 database servers (or fewer with read replicas and caching). Always round up when computing server counts.

Question 14 BSCS AP 31.10.2026

Given the following sequence of integers, construct a Binary Search Tree (BST) by inserting the elements in the order given: 50, 30, 70, 20, 40, 60, 80, 35. Draw the resulting BST and determine its inorder, preorder, and postorder traversal sequences. (10)

BST rule: for every node, all values in the left subtree are smaller and all values in the right subtree are larger. Each new value starts at the root and moves left (smaller) or right (larger) until it finds an empty place.

Insert	Path	Placed as
50	—	root
30	30 < 50 → left	left child of 50
70	70 > 50 → right	right child of 50
20	< 50 → left; < 30 → left	left child of 30
40	< 50 → left; > 30 → right	right child of 30
60	> 50 → right; < 70 → left	left child of 70
80	> 50 → right; > 70 → right	right child of 70
35	< 50 → left; > 30 → right; < 40 → left	left child of 40

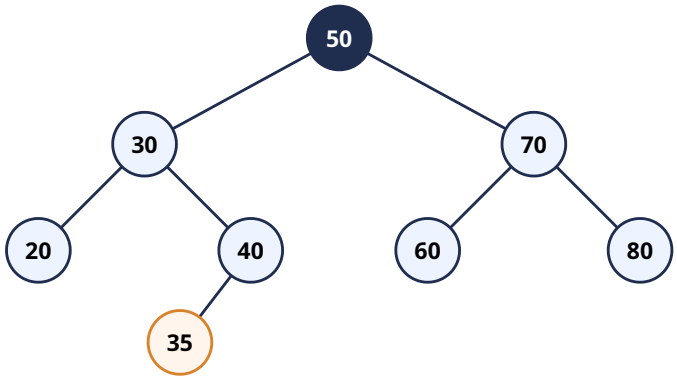


Figure 2.18: The resulting BST (height 3; 35 is the last node inserted)

Traversal	Order	Sequence
Inorder	Left → Root → Right	**20, 30, 35, 40, 50, 60, 70, 80**
Preorder	Root → Left → Right	**50, 30, 20, 40, 35, 70, 60, 80**
Postorder	Left → Right → Root	**20, 35, 40, 30, 60, 80, 70, 50**

Self-check

Inorder of a BST is always sorted ascending — if yours is not, the tree is wrong. Preorder always starts with the root (50); postorder always ends with it.

Question 15 BSCS AP 31.10.2026

A processor has a cache hit ratio of 90%, a cache access time of 5 ns, and a main-memory access time of 80 ns. Calculate the average memory access time, assuming a cache miss requires an additional main-memory access after the cache lookup. (10)

The **Average Memory Access Time (AMAT)** is the expected time for one memory access. Here the cache is always checked first; on a miss (10% of accesses) main memory is accessed *in addition* (a serial, or look-through, access).

AMAT = Hit time + Miss rate × Miss penalty

Solution

Hit ratio h = 0.90 ⇒ miss rate = 1 − 0.90 = 0.10; cache time t_c = 5 ns; memory time t_m = 80 ns.

AMAT = 5 + 0.10 × 80 = 5 + 8 = **13 ns**

Check with the weighted form: h × t_c + (1 − h) × (t_c + t_m) = 0.9 × 5 + 0.1 × 85 = 4.5 + 8.5 = **13 ns** ✓

Interpretation: without the cache every access would take 80 ns; with it the average is 13 ns — about **6.2 times faster** (80 ÷ 13). If the question had said cache and memory are searched in *parallel*, the formula would be 0.9 × 5 + 0.1 × 80 = 12.5 ns — so always read which assumption is given.

Mathematics (Questions 16–19, 5 marks each)

Question 16 BSCS AP 31.10.2026

Pipe A can fill a tank in 10 hours and pipe B in 15 hours. Pipe C can empty the full tank in 12 hours. If all three are opened together, how long will it take to fill the tank? (5)

Solution

In 1 hour: A fills 1/10, B fills 1/15, C empties 1/12 of the tank.

Net part filled in 1 hour = 1/10 + 1/15 − 1/12 = (6 + 4 − 5)/60 = 5/60 = 1/12.

∴ Time to fill the tank = **12 hours**.

LCM method: tank = LCM(10, 15, 12) = 60 units; A = +6, B = +4, C = −5 units/hour ⇒ net 5 units/hour ⇒ 60 ÷ 5 = 12 hours.

Question 17 BSCS AP 31.10.2026

The product of two consecutive positive odd numbers is 143. Find the numbers. (5)

Solution

Let the numbers be x and x + 2.

$x(x + 2) = 143 \Rightarrow x^2 + 2x - 143 = 0 \Rightarrow x^2 + 13x - 11x - 143 = 0 \Rightarrow (x + 13)(x - 11) = 0$

x = 11 or x = −13; the numbers are positive, so x = 11.

∴ The numbers are **11 and 13**. Check: 11 × 13 = 143 ✓

Question 18 BSCS AP 31.10.2026

In a class, if 4 students sit on each bench, then 3 benches remain empty. But if 3 students sit on each bench, then 6 students have to stand. What is the total number of students in the class? (5)

Solution

Let the number of benches be x.
4 per bench with 3 benches empty ⇒ students = 4(x − 3).
3 per bench with 6 standing ⇒ students = 3x + 6.
4(x − 3) = 3x + 6 ⇒ 4x − 12 = 3x + 6 ⇒ x = 18 benches.
Students = 3 × 18 + 6 = **60**. Check: 4 × (18 − 3) = 60 ✓

Question 19 BSCS AP 31.10.2026

The amounts of money held by A, B and C satisfy A:B = 3:4 and B:C = 5:6. If A gives Tk 5,000 to B, the ratio of their amounts becomes 1:2. Find the total amount of money originally held by all three. (5)

Solution

Make B equal in both ratios (LCM of 4 and 5 = 20): A:B = 3:4 = 15:20; B:C = 5:6 = 20:24 ⇒ **A:B:C = 15:20:24**.
Let A = 15k, B = 20k, C = 24k.
After A gives Tk 5,000 to B: (15k − 5,000) : (20k + 5,000) = 1 : 2
⇒ 2(15k − 5,000) = 20k + 5,000 ⇒ 30k − 10,000 = 20k + 5,000 ⇒ 10k = 15,000 ⇒ k = 1,500.
A = 22,500, B = 30,000, C = 36,000.
Total = 59k = 59 × 1,500 = **Tk 88,500**. Check: 17,500 : 35,000 = 1 : 2 ✓

English Proficiency (Questions 20–21)

Question 20 BSCS AP 31.10.2026

Write only the correct answers. (10)

#	Question	Answer	Why
a	By the time we arrived, the film ___ (started / had started / has started)	**had started**	The earlier of two past actions takes the past perfect; "by the time + past" is the signal
b	She is good ___ solving problems.	**at**	Fixed preposition: good at, bad at
c	Neither the teacher nor the students was present.	Neither the teacher nor the students **were** present.	Neither...nor: the verb agrees with the nearer subject (students — plural)
d	Noun form of "decide"	**decision**	decide (v) → decision (n) → decisive (adj)
e	I wish I ___ taller. (am / were / will be)	**were**	Unreal present wish ⇒ past subjunctive "were" for every person

f	Hardly ___ the station when the train left.	**had I reached**	Negative adverb first ⇒ inversion: Hardly + had + subject + V3 … when + past
g	Passive: Someone has stolen my bicycle.	**My bicycle has been stolen.**	Present perfect passive = has been + V3; the vague agent "by someone" is dropped
h	Indirect: She said to me, "Will you help me tomorrow?"	**She asked me if I would help her the next day.**	Yes/no question ⇒ asked + if/whether; will → would; you → I, me → her; tomorrow → the next day
i	Closest in meaning to "meticulous"	**extremely careful**	meticulous = showing great attention to detail
j	Combine with "despite": It was raining heavily. They continued the match.	**Despite the heavy rain, they continued the match.**	despite + noun / gerund (never "despite of" or "despite it was")

Question 21 BSCS AP 31.10.2026

Write an essay on: Job prospects in the age of Artificial Intelligence. (20)

Plan (2 minutes) — what earns 20 marks

Title · introduction with a clear thesis · 3–4 body paragraphs, each with one idea (jobs at risk, jobs created, Bangladesh's situation, skills and policy) · conclusion · about 400–450 words · simple, correct sentences · linking words (however, moreover, therefore, as a result).

Model essay

Job Prospects in the Age of Artificial Intelligence

Introduction. Artificial Intelligence (AI) is no longer a subject of science fiction. Machines can now write reports, answer customers, detect fraud and even generate computer code. This rapid progress has raised a serious question in every household: will AI take away our jobs? The answer is not a simple yes or no. AI will certainly replace some tasks, but it will also create new jobs and change almost every existing one. Our job prospects will depend on how quickly we adapt.

Jobs at risk. AI is most effective at routine and repetitive work. Data entry, basic accounting, call-centre support, simple translation and document checking can now be done faster and cheaper by software. In banks, chatbots already answer balance enquiries, and automated systems verify cheques and KYC documents. Workers whose jobs consist mainly of such tasks face the greatest risk of losing them or of seeing their wages fall.

New opportunities. At the same time, AI is opening doors that did not exist a decade ago. Data scientists, machine-learning engineers, AI trainers, cybersecurity specialists and cloud engineers are in high demand. Even non-technical roles are growing, such as AI product managers, digital marketers and compliance officers who make sure AI is used fairly. According to the World Economic Forum's Future of Jobs Report 2025, technology is expected to create about 170 million jobs and displace about 92 million by 2030 — a net gain, but only for those with the right skills.

The Bangladesh context. For Bangladesh, AI is both a threat and an opportunity. Automation may reduce low-skill jobs in the garment sector and in back-office services. However, the country has a young population and one of the largest freelancing workforces in the world. If our youth learn to use AI tools, they can offer software, design and data services to global clients. Programmes for digital skills, together with the growth of fintech and e-commerce, can turn AI into an engine of employment rather than unemployment.

The way forward. To benefit from AI, we must change how we learn and work. Education should focus on problem-solving, programming, data literacy, communication and creativity — skills that machines cannot easily copy. Workers need lifelong learning through reskilling and upskilling. The government and employers should invest in training, protect workers during the transition, and set rules for the ethical use of AI.

Conclusion. AI will not simply destroy jobs; it will redesign them. People will not be replaced by AI so much as by people who know how to use AI. If we prepare our workforce with the right skills and policies, the age of Artificial Intelligence can become an age of better, safer and more productive jobs for all.

2.5 Paper G — Screening (Preliminary) MCQ Test 2026, Set-3 — complete paper

Item	Information
Exam	বাছাই পরীক্ষা ২০২৬ (screening MCQ test), **Set-3**. The top-right of the header is partly legible and appears to read *পায়রাবন্দর* (Payra Port) — treat the employer as unconfirmed.

Format	100 MCQs, 1 hour, 100 marks; +1 for each correct answer and **−0.50** for each wrong one (as printed)
Mix	IT/CS 33 · Electrical/electronics 7 · English 20 · Bangla 20 · GK 19 · Math 1
What is new	Basic ML theory (ensemble, bias–variance, dummy classifier, hyperparameters), and electrical basics (3-phase, rectifier, RLC resonance) next to standard CS questions

Questions are grouped by topic. Each topic starts with a short background — the concepts, facts and formulas behind its questions — followed by the questions under their original numbers, with the answer and a one-line reason. An answer key in the original order is at the end for timed practice. Bangla-language questions are translated; Bangla terms are kept where the question is about the Bangla word itself. Five questions have a print error or a disputed answer — they are marked ⚠ and explained.

> **With −0.5 per wrong answer**
>
> A blind guess among four options has expected value 0.25 − 0.75 × 0.5 = **−0.125**; after eliminating two options it is +0.25. Guess only when the expected value is positive.

1. C programming basics Q 4, 18, 24, 45, 56, 68

> **Background**
>
> **Data types:** `int` (whole numbers), `float` / `double` (real numbers), `char` (one character). **Operators:** `=` assigns, `==` compares, `%` gives the remainder of integer division (5 % 2 = 1), `!=` not equal. **Statements** end with `;` . **Identifiers** may use letters, digits and `_` , must not start with a digit, must not be a keyword (`int` , `for` …), and may not contain `-` or spaces. **Loops:** `for` and `while` test the condition first (may run 0 times); `do … while` tests after the body, so it runs **at least once**.

4. Which datatype stores whole numbers in C?

(a) float (b) double
(c) char (d) int

Answer: (d) — int stores integers; float/double store real numbers.

18. What is the value of 5%2 in C?

(a) 0 (b) 1
(c) 2 (d) 2.5

Answer: (b) — % is the remainder: 5 = 2×2 + 1.

24. Which symbol is used to end a statement in C/C++?

(a) : (b) ;
(c) , (d) .

Answer: (b) — Semicolon.

45. Which operator is used for assignment in C/C++ programming language?

(a) ==
(b) =
(c) !=
(d) <=

Answer: (b) — = assigns; == compares.

56. Which one is a valid variable name in C?

(a) 2value
(b) int
(c) total-marks
(d) First-name

⚠ Answer: (c) — Strictly, none is valid: 2value starts with a digit, int is a keyword, and '-' is not allowed. The intended answer is (c), a misprint of total_marks.

68. Which loop executes at least once?

(a) for
(b) while
(c) do while
(d) for each

Answer: (c) — do-while tests its condition after the body.

2. Python basics Q 95, 98

Background

Functions are defined with `def name(args):` and an indented body. `for` loops over any iterable; `range(start, stop, step)` produces numbers from start up to *stop* - *1* — e.g. `for i in range(5)` gives 0...4.

95. Which keyword is used to define a function in Python programming language?

(a) function
(b) define
(c) def
(d) func

Answer: (c) — def name(args):

98. Which function is used by the for loop in Python?

(a) couple
(b) range
(c) excuse
(d) control

Answer: (b) — for i in range(n):

3. Data structures and searching Q 13, 69, 73, 83, 99

Background

Stack = LIFO (push/pop at the top: undo, function calls, recursion). **Queue** = FIFO (enqueue at rear, dequeue at front: printer jobs, BFS). **Linear search** checks each item — O(n), works on unsorted data. **Binary search** needs **sorted** data and halves the range each step — O(log n); 1,000,000 items need at most 20 comparisons. A **palindrome** reads the same backwards (madam, 1234321) — checked with two pointers or a stack.

13. A Stack is called ___.

(a) Last In First Out
(b) First In Last Out
(c) Last In Last Out
(d) First In First Out

⚠ Answer: (a) — LIFO is the standard name. FILO means the same thing, but mark LIFO.

69. Which data structure follows the FIFO principle?

(a) Stack
(b) Queue
(c) Tree
(d) Graph

Answer: (b) — Queue.

73. Which searching algorithm requires sorted data?

(a) Linear
(b) Binary
(c) Sequential
(d) Hash

Answer: (b) — Binary search halves a sorted range each step.

83. What is the time complexity of binary search?

(a) O(n)
(b) $O(n^2)$
(c) O(log n)
(d) O(1)

Answer: (c) — O(log n).

99. Which one is an example of a palindrome?

(a) Argentina
(b) Brazil
(c) 1234321
(d) 1024

Answer: (c) — Reads the same backwards.

4. Databases and normalization Q 14, 47, 52

Background

An **RDBMS** stores data in related tables (MySQL, PostgreSQL, Oracle, SQL Server). **Normalization** removes redundancy: 1NF atomic values → 2NF no partial dependency on part of a key → 3NF no transitive dependency → **BCNF**: every determinant is a candidate key. Standard SQL is **declarative** and not Turing-complete (no general loops); recursive CTEs/PL-SQL extend it.

14. What is the use of BCNF?

(a) Database normalization
(b) Fine tuning of software quality
(c) Indexing mechanism
(d) Hashing

Answer: (a) — Boyce–Codd Normal Form: every determinant must be a candidate key.

47. Which one is an example of non-turing machine type of language?

(a) Python
(b) Java
(c) SQL
(d) C++

Answer: (c) — Standard (declarative) SQL is not Turing complete; the others are general-purpose languages.

52. Which is a relational database management system (RDBMS)?

(a) MySQL (b) Python
(c) Java (d) Linux

Answer: (a) — MySQL; Python and Java are languages, Linux is an OS.

5. Machine learning and AI basics Q 3, 20, 50, 58, 76, 93, 94

Background

Idea	Key point
Supervised vs unsupervised	Labelled data (SVM, KNN, linear/logistic regression) vs unlabelled (K-means, PCA)
Ensemble	Combine many models: bagging (Random Forest), boosting (AdaBoost, XGBoost)
Deep learning	Multi-layer neural networks: CNN (images), RNN/LSTM, Transformers (LLMs such as OpenAI's ChatGPT)
Hyperparameter	Set before training: learning rate, regularization α, L1/L2; k in KNN (not in linear regression)
Bias vs variance	Low training accuracy = high bias (underfit); big train–test gap = high variance (overfit)
Dummy / baseline	ZeroR predicts the majority class; a real model must beat it

3. Which one is an example of ensemble machine learning model?

(a) Ada Boost (b) SVM
(c) Linear Regression (d) Logistic Regression

Answer: (a) — AdaBoost combines many weak learners (boosting). Random Forest (bagging) is another ensemble.

20. Which one is not a hyperparameter of linear regression?

(a) Learning rate (b) Alpha (α)
(c) Number of neighbors (d) L1/L2 regularization

Answer: (c) — Number of neighbors (k) is a hyperparameter of KNN.

50. Which one is a dummy classifier?

(a) Linear Regression (b) KNN
(c) SVM (d) Zero-R

Answer: (d) — ZeroR always predicts the majority class — used as a baseline.

58. Which organization created ChatGPT?

(a) Google (b) Microsoft
(c) Open AI (d) Meta

Answer: (c) — OpenAI (Nov 2022).

76. Which one is an example of deep-learning model?

(a) SVM
(b) Naive Bayes
(c) Linear Regression
(d) CNN

Answer: (d) — Convolutional Neural Network; the others are classical ML.

93. If a machine learning model shows 56% and 48% training and test accuracies respectively, it has:

(a) Low bias and low variance
(b) High bias and high variance
(c) Low bias and high variance
(d) High bias and low variance

Answer: (d) — Low training accuracy ⇒ high bias (underfitting); small train–test gap (8%) ⇒ low variance.

94. Which one is an unsupervised learning model?

(a) SVM
(b) KNN
(c) Linear Regression
(d) K-means clustering

Answer: (d) — K-means groups unlabelled data.

6. Networking and the web Q 59, 60, 82

Background

URL (Uniform Resource Locator) = scheme://host/path, e.g. https://www.bb.org.bd/about. **HTTP/HTTPS** (80/443) carry web pages; FTP (20/21) files; SMTP (25) sends and POP3 (110)/IMAP (143) receive e-mail. **DHCP** assigns IP address, mask, gateway and DNS server automatically; **DNS** translates names to IP addresses; a **proxy** forwards requests on a client's behalf.

59. Which one of the following is the full form of URL?

(a) Uniform Resource Locator
(b) Uniform Resource Link
(c) Uniform Reference Link
(d) Uniform Record Locator

Answer: (a) — Uniform Resource Locator.

60. Which protocol is primarily used to browse websites?

(a) FTP
(b) HTTP
(c) SMTP
(d) POP3

Answer: (b) — HTTP/HTTPS. FTP = file transfer; SMTP/POP3 = e-mail.

82. Which server is used to assign IP address dynamically while connecting to the internet?

(a) DNS
(b) Proxy
(c) DHCP
(d) All of the above

Answer: (c) — Dynamic Host Configuration Protocol.

7. Number systems and memory Q 28, 44, 77

> **Background**
>
> **Decimal → binary:** divide by 2 and read remainders bottom-up, or sum powers of 2 (10 = 8 + 2 = 1010). **Primary memory** is directly accessed by the CPU: RAM (volatile — lost without power) and ROM (non-volatile). **Secondary storage** is non-volatile: HDD, SSD, DVD, pen drive. Cache sits between CPU and RAM.

28. What is the binary equivalent for decimal 10?

(a) 1000 (b) 1010
(c) 1100 (d) 1110

Answer: (b) — $10 = 8 + 2 = 1010_2$.

44. What type of memory is RAM?

(a) Permanent (b) External
(c) Volatile (d) Optical

Answer: (c) — Contents are lost when power is off.

77. Which one is an example of primary memory?

(a) HDD (b) SSD
(c) RAM (d) DVD

Answer: (c) — RAM; the others are secondary storage.

8. Operating systems Q 22, 78

> **Background**
>
> **Open-source OS:** Linux (Ubuntu, Fedora), Android (AOSP), FreeBSD. **Proprietary:** Windows, macOS, iOS. A browser (Firefox) is an application, not an OS. Useful Windows shortcuts: Ctrl+Shift+N new folder, Win+E File Explorer, Win+D desktop, Alt+Tab switch, Ctrl+Shift+Esc Task Manager.

22. Which shortcut key is used for creating a new folder in Windows OS?

(a) Ctrl+N (b) Ctrl+Shift+N
(c) Alt+N (d) Ctrl+Alt+N

Answer: (b) — Ctrl+Shift+N in File Explorer; Ctrl+N opens a new window.

78. Which one is the example of open-source operating system?

(a) Windows 11 (b) Mozilla Firefox
(c) iOS (d) Linux

Answer: (d) — Linux. Firefox is open source but a browser, not an OS.

9. Security and software engineering Q 15, 48

Background

Authentication factors: something you know (password, PIN), have (OTP phone, token), are (fingerprint). Two different factors = **2FA/MFA** — strong authentication. AES/RSA are encryption, not authentication processes. **Functional requirements** say what the system does (transfer money); **non-functional** say how well: security, performance, reliability, usability, scalability.

15. Which one is an example of strong authentication process?

(a) AES
(b) RSA
(c) Blockchain
(d) 2FA

Answer: (d) — Two-factor authentication (something you know + have/are). AES and RSA are encryption algorithms.

48. Which one is an example of non-functional properties of a software system?

(a) Coding
(b) Debug
(c) Security
(d) Prototype

Answer: (c) — Non-functional = quality attributes: security, performance, reliability, usability.

10. Electrical and electronics basics Q 26, 34, 75, 81, 86, 89, 91

Background

Ohm's law $V = IR$; power $P = VI = I^2R = V^2/R$. **Passive** components (R, L, C) cannot amplify; **active** ones (transistor, op-amp, SCR) can. Capacitor stores energy in an electric field ($\frac{1}{2}CV^2$); inductor in a magnetic field ($\frac{1}{2}LI^2$). **Power factor** = $\cos\phi$: 1 for pure R, 0 for pure L or C. **Series RLC resonance:** $X_L = X_C \Rightarrow f = 1/(2\pi\sqrt{LC})$, impedance = R (minimum). **3-phase:** phases 120° apart. **Converters:** rectifier AC→DC, inverter DC→AC, transformer changes AC voltage, oscillator generates AC from DC.

26. Which one is a passive electronic component?

(a) Transistor
(b) Operational Amplifier
(c) Resistor
(d) SCR

Answer: (c) — Passive components (R, L, C) cannot amplify or control power; transistor, op-amp and SCR are active.

34. Power factor of a purely resistive circuit is:

(a) 0
(b) 0.5
(c) 1
(d) −1

Answer: (c) — V and I are in phase, $\phi = 0$, $\cos 0 = 1$.

75. In 3-phase system, phase difference between two consecutive phases is:

(a) 60°
(b) 90°
(c) 120°
(d) 180°

Answer: (c) — 360° / 3 = 120°.

81. The device used for AC to DC conversion is:

(a) Inverter
(b) Oscillator
(c) Rectifier
(d) Transformer

Answer: (c) — Rectifier: AC→DC; inverter: DC→AC.

86. If voltage across a 10 Ω resistor is 20 V, its current and power are:

(a) 1 A, 1 W
(b) 2 A, 40 W
(c) 0.5 A, 10 W
(d) 0.5 A, 40 W

Answer: (b) — $I = V/R = 20/10 = 2$ A; $P = VI = 20 \times 2 = 40$ W.

89. A series RLC circuit (R Ω, L H, C F) is driven at frequency f. Which relation is true at resonance?

(a) $f^2 = 1/(4\pi LC)$
(b) $f^2 = 1/(2\pi LC)$
(c) $f = 1/\sqrt{(LC)}$
(d) $f = 1/(2\pi\sqrt{(LC)})$

Answer: (d) — At resonance $X_L = X_C \Rightarrow 2\pi fL = 1/(2\pi fC) \Rightarrow f = 1/(2\pi\sqrt{LC})$.

91. Which element stores electrical energy in an electric field?

(a) inductor
(b) capacitor
(c) transformer
(d) resistor

Answer: (b) — Capacitor ($\frac{1}{2}CV^2$); an inductor stores it in a magnetic field ($\frac{1}{2}LI^2$).

11. English: prepositions and phrasal verbs Q 10, 29, 35, 40, 64

Background

Fixed pairs to memorise: **dream of**, **good at**, **call at** (a place) / **call on** (a person) / **call up** (phone), **stand by** (support), **turn down** (refuse), insist on, depend on, interested in, afraid of, capable of.

10. Faria dreams ___ running her own business.

(a) to
(b) of
(c) by
(d) in

Answer: (b) — dream of/about + gerund.

29. Richard called ___ the bank to deposit a cheque.

(a) in
(b) on
(c) at
(d) up

Answer: (c) — call at = visit a place; call on = visit a person; call up = phone.

35. He is good ___ chess.

(a) in
(b) at
(c) on
(d) with

Answer: (b) — good at + skill.

40. My parents always stand ___ my decision.

(a) with
(b) at
(c) along
(d) by

Answer: (d) — stand by = support.

64. The proposal was too good to ___.

(a) away
(b) down
(c) in
(d) off

⚠ Answer: (b) — The verb is missing in print; the intended phrase is 'too good to turn down' (= refuse).

12. English: verb patterns and special structures Q 16, 32, 67, 79, 80, 97

Background

Structure	Rule	Example
mind / look forward to / object to	+ gerund (V-ing)	Would you mind checking...; I look forward to hearing...
prefer A to B	gerund + to + gerund	She prefers dancing to singing.
No sooner ... than	No sooner had + S + V3 ... than + past	No sooner had I run it than it failed.
as if / as though (earlier unreal)	+ past perfect	He looked as if he had rolled in mud.
It is (high) time	+ S + past simple	It is high time he changed.

16. Will you mind ___ the accounts one more time?

(a) check
(b) checked
(c) to check
(d) checking

Answer: (d) — mind + gerund.

32. Identify the correct sentence:

(a) She prefers dancing to singing.
(b) She prefers dancing than singing.
(c) She prefers to dance than singing.
(d) She prefers dancing over to sing.

Answer: (a) — prefer + gerund + to + gerund.

67. No sooner had I run the program than the system ___.

(a) had failed
(b) fail
(c) would fail
(d) failed

Answer: (d) — No sooner + had + V3 ... than + past simple.

79. I look forward to ___ from you.

(a) hearing
(b) heard
(c) hear
(d) have heard

Answer: (a) — 'to' here is a preposition → gerund.

80. The boy was so dirty that he looked as if he ___ in the mud.

(a) rolled
(b) has rolled
(c) had rolled
(d) is rolling

Answer: (c) — as if + past perfect for an earlier unreal action.

97. It is high time he (change) his behaviour.

(a) change
(b) has changed
(c) changed
(d) did change

Answer: (c) — It is high time + subject + past simple.

13. English: parts of speech and word forms Q 19, 55, 66

Background

A word that describes a noun, or follows a linking verb (be, seem, look) to describe the subject, is an **adjective** (hereditary, disagreeable). Common adjective suffixes: -ous, -ful, -able/-ible, -ive, -al, -ed (various, variable, varied). Not every suffix fits every root — 'variful' does not exist.

19. Depression is often hereditary. Here, the underlined word (hereditary) is a/an ___.

(a) noun
(b) adverb
(c) adjective
(d) verb

Answer: (c) — It describes the noun 'depression' after the linking verb 'is'.

55. Karim is a rather disagreeable man. Here, the underlined word (disagreeable) is a/an ___.

(a) noun
(b) verb
(c) adjective
(d) adverb

Answer: (c) — It qualifies the noun 'man'.

66. Which one is not an acceptable adjective form of the word "vary"?

(a) various
(b) variful
(c) variable
(d) varied

Answer: (b) — 'Variful' is not an English word.

14. English: vocabulary, idioms, spelling, literature Q 9, 27, 30, 36, 51, 53

Background

Occupation words: curator (museum), pallbearer (coffin), janitor (cleaning), archivist (records). Idioms: through thick and thin (in all circumstances), once in a blue moon (rarely), at sixes and sevens (in disorder). Spelling traps: diarrhoea, accommodation, committee, separate, occurrence. Romantic poets: Keats (To Autumn, Ode to a Nightingale), Wordsworth (Daffodils), Shelley (Ode to the West Wind).

9. The synonym of "commemorate" is ___.

(a) observe
(b) memorise
(c) forget
(d) overlook

Answer: (a) — Commemorate = honour the memory of / observe (a day). Memorise means learn by heart.

27. "Through thick and thin" means:

(a) by chance
(b) very quickly
(c) in all circumstances
(d) abruptly

Answer: (c) — In good times and bad — under all circumstances.

30. "To Autumn" is written by ___.

(a) William Shakespeare
(b) William Wordsworth
(c) John Keats
(d) Robert Frost

Answer: (c) — John Keats, ode of 1819.

36. Which one of the following is correctly spelt?

(a) Dirohea
(b) Diarrhoea
(c) Dirrohea
(d) Dirrhoea

Answer: (b) — Diarrhoea (British) / diarrhea (American).

51. A person in charge of a museum is a/an ___.

(a) Librarian
(b) architect
(c) curator
(d) janitor

Answer: (c) — Curator.

53. The people who carry a coffin at a funeral are called ___.

(a) mourners
(b) pallbearers
(c) grave diggers
(d) processioners

Answer: (b) — Pallbearers.

15. Bangla literature Q 1, 12, 37, 41, 42, 46, 63, 65, 72, 88, 92

Background

Periods: ancient (Charyapada, found 1907 by Haraprasad Shastri) → medieval (Srikrishnakirtan by Baru Chandidas, found 1909; Abdul Hakim's 'বঙ্গবাণী'; last poet Bharatchandra, d. 1760) → modern (from 1800, Fort William College starts Bangla prose; Vidyasagar's বর্ণপরিচয় 1855). **Must-know authors:** Rokeya (অবরোধবাসিনী, মতিচূর), Sarat Chandra (পথের দাবী — banned), Bibhutibhushan (পথের পাঁচালী → আম আঁটির ভেঁপু), Pramatha Chowdhury (pen name বীরবল), Rabindranath (ডাকঘর — drama). **Liberation War novels:** রাইফেল রোটি আওরাত (Anwar Pasha), জোছনা ও জননীর গল্প (Humayun Ahmed), হাঙর নদী গ্রেনেড (Selina Hossain).

1. Which one is a work of Begum Rokeya Sakhawat Hossain?

(a) Nari
(b) Abarodhbasini
(c) Sanskriti-Katha
(d) Sanjher Maya

Answer: (b) — Abarodhbasini (1931) — also Motichur, Padmarag, Sultana's Dream. Sanjher Maya is by Sufia Kamal; Sanskriti-Katha by Motaher Hossain Chowdhury.

12. Which institution played a role in the development of Bangla prose?

(a) Hindu College
(b) Sanskrit College
(c) Fort William College
(d) Calcutta College

Answer: (c) — Fort William College (1800), where William Carey's pundits wrote the first Bangla prose textbooks.

37. Who wrote the poem 'বঙ্গবাণী' (Bangabani)?

(a) Abdul Hakim
(b) Michael Madhusudan Dutt
(c) Kazi Nazrul Islam
(d) Jibanananda Das

Answer: (a) — Abdul Hakim (17th c.): 'যে সবে বঙ্গেত জন্মি হিংসে বঙ্গবাণী...'.

41. Who wrote 'বর্ণপরিচয়' (Barnaparichay)?

(a) Bankim Chandra Chattopadhyay
(b) Raja Rammohan Roy
(c) Ishwar Chandra Vidyasagar
(d) William Carey

Answer: (c) — Ishwar Chandra Vidyasagar (1855).

42. When was 'শ্রীকৃষ্ণকীর্তন' (Srikrishnakirtan) discovered?

(a) 1902
(b) 1907
(c) 1909
(d) 1911

Answer: (c) — Found by Basanta Ranjan Roy in 1909 (Bankura); published 1916. (Charyapada: 1907.)

46. The story 'আম আঁটির ভেঁপু' is related to which novel?

(a) Kapalkundala
(b) Pather Panchali
(c) Putul Nacher Itikatha
(d) Kabi

Answer: (b) — Abridged children's edition of Bibhutibhushan's Pather Panchali.

63. Which novel of Sarat Chandra Chattopadhyay was banned by the British government?

(a) Debdas
(b) Pather Dabi
(c) Charitraheen
(d) Grihadaha

Answer: (b) — Pather Dabi (1926), banned for its revolutionary theme.

65. Who is the last poet of the medieval period of Bangla literature?

(a) Bharatchandra Ray Gunakar
(b) Shah Muhammad Sagir
(c) Chandidas
(d) Ishwar Chandra Gupta

Answer: (a) — Bharatchandra (d. 1760). Ishwar Gupta is the 'poet of the transition age' (যুগসন্ধিক্ষণের কবি).

72. 'বীরবল' (Birbal) is the pen name of whom?

(a) Biharilal Chakraborty
(b) Rabindranath Tagore
(c) Syed Mujtaba Ali
(d) Pramatha Chowdhury

Answer: (d) — Pramatha Chowdhury (editor of Sabuj Patra).

88. Which is a novel based on the Liberation War?

(a) Arek Falgun
(b) Raifel Roti Awrat
(c) Khoabnama
(d) Kritadasher Hasi

Answer: (b) — Anwar Pasha (1973). Arek Falgun = Language Movement; Khoabnama = Tebhaga/Pakistan era; Kritadasher Hasi = allegory of Ayub rule.

92. What type of work is Rabindranath's 'ডাকঘর' (Dakghar)?

(a) short story
(b) novel
(c) drama
(d) essay

Answer: (c) — Drama (1912).

16. Bangla grammar Q 5, 25, 31, 49, 62, 70, 90, 96, 100

Background

Branches: ধ্বনিতত্ত্ব (phonology), রূপতত্ত্ব (morphology), বাক্যতত্ত্ব (syntax), অর্থতত্ত্ব (semantics). Basic vowel sounds (মৌলিক স্বরধ্বনি) = 7: অ আ ই উ এ ও অ্যা. **Pronunciation:** ক্ষ = ক্‌খ, and initial অ before a conjunct often sounds ও (দক্ষ → দোক্‌খো). **উপমান কর্মধারয়:** standard + common quality (ভ্রমরের ন্যায় কালো = ভ্রমরকালো); উপমিত: compared object + standard (মুখ চন্দ্রের ন্যায় = মুখচন্দ্র). **Spelling:** দুরবস্থা, উপর্যুক্ত, শুধু/মাত্র (not both). **One-word:** যা নিবারণ করা যায় না — অনিবার্য; কষ্টে নিবারণ করা যায় — দুর্নিবার. Idioms: আমড়া কাঠের ঢেঁকি — অপদার্থ.

5. Which spelling is correct?

(a) উপরোক্ত
(b) দুরবস্থা
(c) দুরাবস্থা
(d) শুধুমাত্র

Answer: (b) — দুরবস্থা (দুঃ + অবস্থা) is correct. Correct forms of the others: উপর্যুক্ত, দুরবস্থা, শুধু / মাত্র.

25. What does the Bangla proverb 'নদীর পাড়ের গাছ' (a tree on the river bank) mean?

(a) dangerous
(b) small
(c) uncertain existence
(d) short-lived

⚠ Answer: (c) — A tree on an eroding river bank may fall at any time — its existence is uncertain.

31. What does the idiom 'আমড়া কাঠের ঢেঁকি' mean?

(a) unscrupulous
(b) worthless person
(c) rare thing
(d) forcible push

Answer: (b) — Worthless / good-for-nothing (অপদার্থ).

49. What is the correct pronunciation of 'দক্ষ'?

(a) দোক্‌খো
(b) দোক্‌খ
(c) দক্‌খো
(d) দক্‌কো

Answer: (a) — ক্ষ is pronounced ক্‌খ and the first অ becomes ও: দোক্‌খো.

62. What is the Bangla term for 'Phonology'?

(a) ধ্বনিতত্ত্ব
(b) রূপতত্ত্ব
(c) বাক্যতত্ত্ব
(d) অর্থতত্ত্ব

Answer: (a) — Phonology = ধ্বনিতত্ত্ব; morphology = রূপতত্ত্ব; syntax = বাক্যতত্ত্ব; semantics = অর্থতত্ত্ব.

70. Which compound (সমাস) is 'ভ্রমর কালো'?

(a) উপমান কর্মধারয়
(b) সাধারণ কর্মধারয়
(c) উপমিত কর্মধারয়
(d) মধ্যপদলোপী কর্মধারয়

Answer: (a) — ভ্রমরের ন্যায় কালো — the standard of comparison (ভ্রমর) + common quality (কালো) = upaman karmadharay.

90. How many basic (মৌলিক) vowel sounds are there in Bangla?

(a) 11
(b) 9
(c) 7
(d) 5

Answer: (c) — অ, আ, ই, উ, এ, ও, অ্যা — 7. (Vowel letters: 11.)

96. Which word is a synonym of 'মতি'?

(a) মনন
(b) বৃদ্ধি
(c) রুচি
(d) অলংকার

Answer: (c) — মতি = মন, বুদ্ধি, রুচি, প্রবৃত্তি. Note (b) is বৃদ্ধি (growth), not বুদ্ধি.

100. One word for 'যা নিবারণ করা যায় না' (that cannot be prevented):

(a) দুর্নিবার
(b) অনিবার্য
(c) দুর্লভ
(d) দুর্দমনীয়

Answer: (b) — অনিবার্য. দুর্নিবার = that is hard to prevent; দুর্দমনীয় = hard to suppress.

17. GK: Bangladesh Q 2, 7, 8, 11

Background

Seaports: Chattogram (largest, 1888), Mongla (2nd, 1950), Payra (3rd, 2013); Matarbari deep-sea port under construction. **Gas fields:** Titas largest by reserve, Bibiyana largest producer; first found: Haripur (1955). **Elections:** first parliamentary election 7 March 1973. **Rivers:** Padma, Meghna, Jamuna are the main systems; guidebooks usually list Padma as flowing through the most districts.

2. Which is the second largest seaport of Bangladesh?

(a) Payra Port
(b) Mongla Port
(c) Chattogram Port
(d) Matarbari Port

Answer: (b) — Chattogram is the largest, Mongla second, Payra third.

7. By reserves, which is the largest gas field of Bangladesh?

(a) Sangu
(b) Kailashtila
(c) Meghna
(d) Titas

Answer: (d) — Titas (Brahmanbaria) is the largest by reserve; Bibiyana is the largest by current production (not an option).

8. Which river of Bangladesh flows through the most districts?

(a) Padma
(b) Jamuna
(c) Meghna
(d) Mahananda

⚠ Answer: (a) — Padma — the answer given in most job-exam guides (sources differ; some say Meghna).

11. In which year was the first parliamentary election of Bangladesh held after independence?

(a) 1971
(b) 1972
(c) 1973
(d) 1974

Answer: (c) — 7 March 1973.

18. GK: international affairs Q 6, 17, 21, 23, 33, 38, 43, 54, 57, 61, 71, 74, 84, 85, 87

Background

UN: Charter signed 26 June 1945, in force 24 October 1945 (UN Day); HQ New York — also UNICEF, UNDP; Geneva hosts WHO, ILO, UNHCR; Paris UNESCO; Rome FAO. Smallest member by area: Monaco; Vatican City is an observer. **Treaties:** Kyoto Protocol (1997, greenhouse gases), Paris Agreement (2015), CEDAW (1979, women's rights); COP-31 in 2026 is hosted by Türkiye. **Others:** Brazil largest in South America; Amazon = lungs of the earth; Mars = red planet; Churchill = WWII PM; Chancellor heads the government in Germany/Austria; persona non grata = unwelcome diplomat; BIMSTEC chair: Bangladesh (from April 2025); FIFA World Cup 2030 Morocco–Spain–Portugal, 2034 Saudi Arabia.

6. Which is the largest country of South America by area?

(a) Argentina
(b) Brazil
(c) Peru
(d) Colombia

Answer: (b) — Brazil (≈8.5 million km², about half of the continent).

17. The Kyoto Protocol is about ___.

(a) climate change
(b) nuclear non-proliferation
(c) preventing desertification
(d) sharing water of international rivers

Answer: (a) — 1997 treaty to cut greenhouse-gas emissions (UNFCCC).

21. Where is the headquarters of UNICEF?

(a) New York
(b) London
(c) Geneva
(d) Paris

Answer: (a) — New York, USA.

23. Where is the headquarters of the UN agency UNDP?

(a) Toronto, Canada
(b) Amsterdam, Netherlands
(c) London, UK
(d) New York, USA

Answer: (d) — New York, USA.

33. Where will COP-31 be held?

(a) Türkiye
(b) Mexico
(c) Sydney
(d) Berlin

Answer: (a) — COP-31 (November 2026) is hosted in Antalya, Türkiye, with Australia leading the negotiations.

38. Which country will host the 2034 FIFA World Cup?

(a) Russia
(b) Saudi Arabia
(c) Morocco
(d) Spain

Answer: (b) — Saudi Arabia (confirmed Dec 2024). 2030: Morocco–Spain–Portugal.

43. By area, which is the smallest member state of the UN among these?

(a) Tuvalu
(b) Haiti
(c) Vatican City
(d) Nauru

Answer: (d) — Vatican City is only an observer, not a member. Nauru (21 km²) < Tuvalu (26 km²). (Monaco, 2 km², is the true smallest member.)

54. Who was the Prime Minister of the UK during the Second World War?

(a) Winston Churchill
(b) Harold Macmillan
(c) Anthony Eden
(d) Margaret Thatcher

Answer: (a) — Churchill (1940–45). Neville Chamberlain was PM at the start (1939–40) but is not an option.

57. Which planet is called the red planet?

(a) Mars
(b) Mercury
(c) Jupiter
(d) Venus

Answer: (a) — Mars (iron oxide on its surface).

61. The CEDAW convention speaks of whose rights?

(a) children
(b) refugees
(c) migrants
(d) women

Answer: (d) — Convention on the Elimination of All Forms of Discrimination Against Women (1979).

71. From which date did the UN Charter come into force?

(a) 26 June 1945
(b) 24 October 1945
(c) 10 December 1945
(d) 10 January 1946

Answer: (b) — Signed 26 June 1945; in force 24 October 1945 (UN Day).

74. Which of these is called the lungs of the earth?

(a) Sundarbans
(b) Amazon rainforest
(c) Boreal forest
(d) Congo rainforest

Answer: (b) — Amazon rainforest.

84. What does 'Persona non grata' mean?

(a) honoured person
(b) ambassador-level person
(c) unacceptable person
(d) minister-level person

Answer: (c) — A diplomat declared unwelcome by the host country.

85. Which country currently holds the BIMSTEC chairmanship?

(a) Thailand
(b) India
(c) Nepal
(d) Bangladesh

Answer: (d) — Bangladesh took over from Thailand at the 6th summit (Bangkok, April 2025).

87. The Chancellor is the head of government of which country?

(a) Canada
(b) France
(c) Poland
(d) Germany

Answer: (d) — Germany (also Austria).

19. Math: sets Q 39

Background

Inclusion–exclusion: $|A \cup B| = |A| + |B| - |A \cap B|$; for three sets add back $|A \cap B \cap C|$: $|A \cup B \cup C| = \Sigma A - \Sigma(\text{pairs}) + |A \cap B \cap C|$. Example: 40 like tea, 30 coffee, 10 both → 60 like at least one.

39. Which one is the principle of inclusion/exclusion of two sets intersecting each other?

(a) $|A \cup B| = |A| + |B| - |A \cap B|$ (b) $|A \cup B| = |A| + |B|$

(c) $|A \cap B| = |A| + |B|$ (d) $|A \cap B| = |A| - |B| + |A \cup B|$

Answer: (a) — Elements of $A \cap B$ are counted twice in $|A| + |B|$, so subtract once.

Answer key (original order)

1–b	2–b	3–a	4–d	5–b	6–b	7–d	8–a	9–a	10–b
11–c	12–c	13–a	14–a	15–d	16–d	17–a	18–b	19–c	20–c
21–a	22–b	23–d	24–b	25–c	26–c	27–c	28–b	29–c	30–c
31–b	32–a	33–a	34–c	35–b	36–b	37–a	38–b	39–a	40–d
41–c	42–c	43–d	44–c	45–b	46–b	47–c	48–c	49–a	50–d
51–c	52–a	53–b	54–a	55–c	56–c	57–a	58–c	59–a	60–b
61–d	62–a	63–b	64–b	65–a	66–b	67–d	68–c	69–b	70–a
71–b	72–d	73–b	74–b	75–c	76–d	77–c	78–d	79–a	80–c
81–c	82–c	83–c	84–c	85–d	86–b	87–d	88–b	89–d	90–c
91–b	92–c	93–d	94–d	95–c	96–c	97–c	98–b	99–c	100–b

Chapter 3 — Programming Problem Bank

In every BIBM AP written exam, 2–3 programs had to be written: primes from 1 to n, series sum (1+2+4+7+11...), Floyd/pattern triangle, LCM, smallest value of an array, weekly pay, recursive factorial, insertion sort, and output tracing (recursion). This chapter has 46 programs including these, and 21 output-tracing problems. Every code listing is compilable in C99 or Java 8+; if the question says "C/Java", writing in either one is enough.

3.1 Rules for Writing Code on Paper and Getting Full Marks

6 steps (8–10 minutes per program)

1. Write the **header and main** in full (`#include <stdio.h>` , `int main(void)` , `return 0;`) — if the examiner sees half-written code, marks are cut.
2. **Input prompt + scanf/Scanner** — if the question says "n is given by user input", you must take input; do not hard-code.
3. **Validation** in one line: `if (n < 1) { printf("Invalid"); return 1; }`
4. **Logic in a separate function** (e.g. `isPrime()`) — easier to read, helps in partial marking.
5. **2–3 comments** — show where the logic is (`/* divide up to √n */`).
6. Write **sample I/O and complexity** below the code: *Input: 20 → Output: 2 3 5 7 11 13 17 19; Time O(n√n).*

Common mistakes that lose marks

`scanf("%d", n)` (missing &), `1/i` integer division (should be `1.0/i`), wrong `<=` vs `<` in a loop, `=` vs `==` , `int` overflow in factorial ($13! > 2^{31}$), not leaving space for `'\0'` at the end of a string, using `==` to compare Strings in Java.

3.2 Number-Based Programs

P1. All primes from 1 to n AP 2024 (10198)

Write a C/Java program that shows prime numbers between 1 to n; n is given by user input.

```c
#include <stdio.h>

int isPrime(int n){
    if (n < 2) return 0;                 /* 0, 1 are not prime */
    for (int i = 2; i * i <= n; i++)     /* dividing up to √n is enough */
        if (n % i == 0) return 0;
    return 1;
}

int main(void){
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    if (n < 2) { printf("No prime\n"); return 0; }
    for (int i = 1; i <= n; i++)
        if (isPrime(i)) printf("%d ", i);
    printf("\n");
    return 0;
}
```

```java
// Java
import java.util.Scanner;
public class Primes {
    static boolean isPrime(int n){
        if (n < 2) return false;
        for (int i = 2; (long) i * i <= n; i++)
            if (n % i == 0) return false;
        return true;
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for (int i = 1; i <= n; i++)
            if (isPrime(i)) System.out.print(i + " ");
    }
}
```

Input: 20 → Output: 2 3 5 7 11 13 17 19 · Time O(n√n), Space O(1)

P2. Sieve of Eratosthenes (fast for large n)

```
#include <stdio.h>
#define MAX 1000000
static char comp[MAX + 1];              /* comp[i] = 1 means composite */

int main(void){
    int n;
    scanf("%d", &n);                    /* assume n <= MAX */
    for (int i = 2; (long long) i * i <= n; i++)
        if (!comp[i])
            for (int j = i * i; j <= n; j += i) comp[j] = 1;
    for (int i = 2; i <= n; i++)
        if (!comp[i]) printf("%d ", i);
    return 0;
}
```

Input: 30 → 2 3 5 7 11 13 17 19 23 29 · Time O(n log log n), Space O(n). Start from i×i, because the smaller multiples were already crossed out by earlier primes.

P3. GCD and LCM BB AP 2023

Given two integers A and B, write a program to compute LCM of A and B. Relation: **LCM × GCD = A × B**.

```
#include <stdio.h>

int gcd(int a, int b){                  /* Euclid: gcd(a,b) = gcd(b, a % b) */
    while (b != 0) { int r = a % b; a = b; b = r; }
    return a;
}

int main(void){
    int a, b;
    printf("Enter A and B: ");
    scanf("%d %d", &a, &b);
    if (a <= 0 || b <= 0) { printf("Enter positive integers\n"); return 1; }
    int g = gcd(a, b);
    long long lcm = (long long) a / g * b;   /* divide first, then multiply — avoids
overflow */
    printf("GCD = %d, LCM = %lld\n", g, lcm);
    return 0;
}
```

Step (12, 18)	a	b	r = a % b
1	12	18	12
2	18	12	6
3	12	6	0
End	6	0	—

Output: GCD = 6, LCM = 36 · Time O(log min(a,b)) · Python: `import math; g = math.gcd(a, b); print(a * b // g)`

P4. Factorial (iterative)

```
unsigned long long fact = 1;               /* holds up to 20! */
for (int i = 2; i <= n; i++) fact *= i;
printf("%d! = %llu\n", n, fact);
```

Input 5 → 5! = 120 · O(n). The recursive version is in P36.

P5. First n terms of Fibonacci

```
long long a = 0, b = 1;
for (int i = 1; i <= n; i++) {
    printf("%lld ", a);
    long long t = a + b;  a = b;  b = t;
}
```

Input 8 → 0 1 1 2 3 5 8 13 · O(n)

P6. Armstrong number (153 = $1^3 + 5^3 + 3^3$)

```
int isArmstrong(int n){
    int k = 0, sum = 0;
    for (int t = n; t > 0; t /= 10) k++;          /* number of digits */
    for (int t = n; t > 0; t /= 10) {
        int d = t % 10, p = 1;
        for (int i = 0; i < k; i++) p *= d;        /* d^k */
        sum += p;
    }
    return sum == n;
}
/* main: for (i = 1; i <= 1000; i++) if (isArmstrong(i)) printf("%d ", i); */
```

1–1000: 1 2 3 4 5 6 7 8 9 153 370 371 407 · O(d^2) per number (d = number of digits)

P7. Perfect number (sum of proper divisors = the number)

```
int isPerfect(int n){
    if (n < 2) return 0;
    int sum = 1;                                  /* 1 is always a divisor */
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) {
            sum += i;
            if (i != n / i) sum += n / i;         /* the other divisor of the pair */
        }
    return sum == n;
}
```

6 = 1+2+3, 28 = 1+2+4+7+14, 496 → Perfect · O(√n)

P8. Reverse a number, palindrome number and digit sum (all together)

```
int n, t, rev = 0, sum = 0;
scanf("%d", &n);
for (t = n; t > 0; t /= 10) {
    int d = t % 10;          /* last digit */
    rev = rev * 10 + d;
    sum += d;
}
printf("Reverse = %d, Digit sum = %d, %s\n", rev, sum,
       rev == n ? "Palindrome" : "Not palindrome");
```

12321 → Reverse = 12321, Digit sum = 9, Palindrome · 1234 → 4321, 10, Not palindrome · O(d)

P9. Decimal ↔ Binary BSCS Officer IT 2020

```
void toBinary(int n){                /* recursion: prints the higher bits first */
    if (n > 1) toBinary(n / 2);
    printf("%d", n % 2);
}
int toDecimal(const char *s){        /* "1101" → 13 */
    int v = 0;
    for (int i = 0; s[i] != '\0'; i++) v = v * 2 + (s[i] - '0');
    return v;
}
```

toBinary(13) → 1101 · toDecimal("101101") → 45 · O(log n)

Manual conversion (2020 question)

(i) $(10010.101)_2$ = 16 + 2 + 0.5 + 0.125 = **$(18.625)_{10}$**

(ii) $(543)_{10}$ → 543 ÷ 16 = 33 remainder 15 (F); 33 ÷ 16 = 2 remainder 1; 2 ÷ 16 = 0 remainder 2 → read bottom to top: **$(21F)_{16}$**. Check: 2×256 + 1×16 + 15 = 543 ✔

P10. Power x^n — fast method (Exponentiation by squaring)

```
long long power(long long x, int n){     /* O(log n) */
    long long r = 1;
    while (n > 0) {
        if (n & 1) r *= x;                  /* if the bit is odd, multiply */
        x *= x;
        n >>= 1;
    }
    return r;
}
```

power(2, 10): n=10→(x=4,n=5)→(r=4,x=16,n=2)→(x=256,n=1)→(r=1024) → 1024

P11. Leap year

```
int isLeap(int y){ return (y % 4 == 0 && y % 100 != 0) || (y % 400 == 0); }
```

2024 → Leap, 1900 → Not leap (divisible by 100 but not 400), 2000 → Leap

P12. Swap without a temporary variable

```
a = a + b;  b = a - b;  a = a - b;      /* risk of overflow */
/* or */
a ^= b;     b ^= a;     a ^= b;         /* XOR: if a and b are the same memory (the same
variable), both become 0 */
```

a = 5, b = 9 → a = 9, b = 5

P13. Weekly pay BSCS Officer IT 2020

Total weekly pay = hourly wage × regular hours + overtime hours × 1.5 × hourly wage.

```
#include <stdio.h>
int main(void){
    double wage, regular, overtime;
    printf("Hourly wage, regular hours, overtime hours: ");
    scanf("%lf %lf %lf", &wage, &regular, &overtime);
    if (wage < 0 || regular < 0 || overtime < 0) { printf("Invalid input\n"); return 1; }
    double pay = wage * regular + overtime * 1.5 * wage;
    printf("Total weekly pay = %.2f\n", pay);
    return 0;
}
```

```
# Python
w, r, o = map(float, input().split())
print(f"Total weekly pay = {w * r + o * 1.5 * w:.2f}")
```

Input: 200 40 5 → 200×40 + 5×300 = 8000 + 1500 → Total weekly pay = 9500.00

P14. Largest of three numbers — flowchart and code BSCS AP 17.01.2026

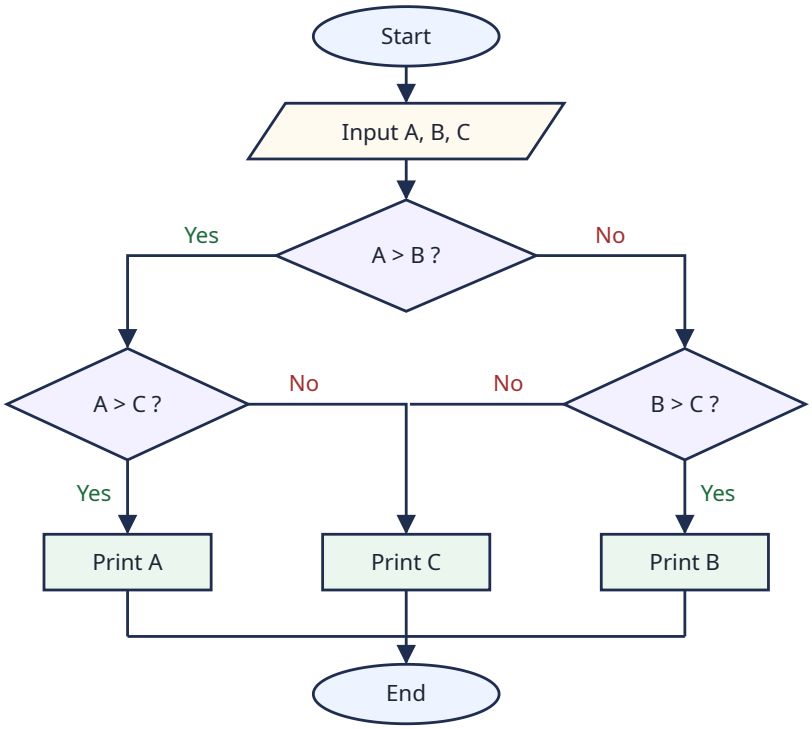


Figure 3.1: Flowchart to find the largest of three distinct numbers

```
if (a > b) { if (a > c) max = a; else max = c; }
else       { if (b > c) max = b; else max = c; }
/* in short: max = (a > b) ? (a > c ? a : c) : (b > c ? b : c); */
```

Input: 12 45 7 → 45 · always exactly 2 comparisons

3.3 Series Sum

P15. 1 + 2 + 4 + 7 + 11 + … up to the n-th term AP 2024 (10198)

Pattern: the difference grows 1, 2, 3, 4, … The k-th term $t_k = t_{k-1} + (k - 1) = 1 + k(k - 1)/2$.

$$S_n = \Sigma\,[1 + k(k - 1)/2] = n + (n - 1)n(n + 1)/6 \quad (n = 5\text{: } 5 + 4{\cdot}5{\cdot}6/6 = 25)$$

```
#include <stdio.h>
int main(void){
    int n;
    printf("Enter number of terms: ");
    scanf("%d", &n);
    long long term = 1, sum = 0;
    for (int k = 1; k <= n; k++) {
        sum += term;
        printf("%lld%s", term, k < n ? " + " : " = ");
        term += k;                    /* next term = this term + k */
    }
    printf("%lld\n", sum);
    return 0;
}
```

```
// Java
import java.util.Scanner;
public class Series {
    public static void main(String[] args){
        int n = new Scanner(System.in).nextInt();
        long term = 1, sum = 0;
        for (int k = 1; k <= n; k++) {
            sum += term;
            System.out.print(term + (k < n ? " + " : " = "));
            term += k;
        }
        System.out.println(sum);
    }
}
```

Input 5 → 1 + 2 + 4 + 7 + 11 = 25 · O(n) (O(1) with the formula)

P16. 1 + 1/2 + 1/3 + … + 1/n (Harmonic)

```
double sum = 0.0;
for (int i = 1; i <= n; i++) sum += 1.0 / i;    /* writing 1/i gives integer division → 0
*/
printf("%.6f\n", sum);
```

n = 4 → 2.083333

P17. $1^2 + 2^2 + \ldots + n^2$

```
long long sum = 0;
for (int i = 1; i <= n; i++) sum += (long long) i * i;
/* check: n(n+1)(2n+1)/6 */
```

n = 5 → 55

P18. $1 - x^2/2! + x^4/4! - \ldots$ (cos x series, n terms)

Each term from the previous term: $t_{k+1} = -t_k \times x^2 / ((2k-1)(2k))$ — no separate factorial needed.

```
#include <stdio.h>
int main(void){
    double x; int n;
    scanf("%lf %d", &x, &n);              /* x in radians, n = number of terms */
    double term = 1.0, sum = 0.0;
    for (int k = 1; k <= n; k++) {
        sum += term;
        term = -term * x * x / ((2 * k - 1) * (2 * k));
    }
    printf("Sum = %.6f\n", sum);
    return 0;
}
```

Input: 1 5 → 1 − 0.5 + 0.0416667 − 0.0013889 + 0.0000248 → Sum = 0.540303 (cos 1 = 0.540302) · O(n)

3.4 Patterns

> **Formula for patterns**
>
> Outer loop = row (i), inner loop = column (j). Write how many spaces and how many symbols each row has as a formula in i (e.g. space = n − i, star = 2i − 1).

P19. Floyd's triangle (numbers)

```
int num = 1;
for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= i; j++)
        printf("%d ", num++);
    printf("\n");
}
```

```
n = 4:
1
2 3
4 5 6
7 8 9 10
```

P20. Floyd's 0-1 triangle AP 2024 (10198)

For row i and column j (starting from 1), the value is 1 if (i + j) is even, 0 if odd.

```c
#include <stdio.h>
int main(void){
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++)
            printf("%d", (i + j) % 2 == 0
? 1 : 0);
        printf("\n");
    }
    return 0;
}
```

```
n = 5:
1
01
101
0101
10101
```

```java
// Java
import java.util.Scanner;
public class Floyd01 {
    public static void main(String[] args){
        int n = new Scanner(System.in).nextInt();
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= i; j++)
                System.out.print((i + j) % 2 == 0 ? 1 : 0);
            System.out.println();
        }
    }
}
```

O(n²). Check: i=3 → j=1:4 even→1, j=2:5→0, j=3:6→1 ⇒ "101" ✔

P21. Star pyramid

```c
for (int i = 1; i <= n; i++) {
    for (int s = 1; s <= n - i; s++)
printf(" ");
    for (int k = 1; k <= 2 * i - 1; k++)
printf("*");
    printf("\n");
}
```

```
n = 4:
   *
  ***
 *****
*******
```

P22. Pascal's triangle

Next value in row i: C(i, j+1) = C(i, j) × (i − j) / (j + 1).

```
for (int i = 0; i < n; i++) {
    for (int s = 0; s < n - i - 1; s++)
printf(" ");
    int c = 1;
    for (int j = 0; j <= i; j++) {
        printf("%d ", c);
        c = c * (i - j) / (j + 1);
    }
    printf("\n");
}
```

```
n = 5:
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

P23. Diamond

```
void row(int n, int i){
    for (int s = 0; s < n - i; s++)
printf(" ");
    for (int k = 0; k < 2 * i - 1; k++)
printf("*");
    printf("\n");
}
/* main: */
for (int i = 1; i <= n; i++)     row(n,
i);  /* upper half */
for (int i = n - 1; i >= 1; i--) row(n,
i);  /* lower half */
```

```
n = 3:
  *
 ***
*****
 ***
  *
```

3.5 Arrays and Matrices

P24. Smallest value by passing an array to a function Combined 5 Bank AP 2023

```
#include <stdio.h>

int findMin(const int a[], int n){      /* the array decays to a pointer, so n must be
passed separately */
    int min = a[0];
    for (int i = 1; i < n; i++)
        if (a[i] < min) min = a[i];
    return min;
}

int main(void){
    int a[] = {34, 7, 23, 32, 5, 62};
    int n = sizeof(a) / sizeof(a[0]);
    printf("Smallest = %d\n", findMin(a, n));
    return 0;
}
```

```
// Java: length stays with the array
static int findMin(int[] a){
    int min = a[0];
    for (int x : a) if (x < min) min = x;
    return min;
}
```

Output: Smallest = 5 · Time O(n), Space O(1). For Max, just change `<` → `>`.

P25. Second largest (single traversal)

```
#include <limits.h>
int first = INT_MIN, second = INT_MIN;
for (int i = 0; i < n; i++) {
    if (a[i] > first) { second = first; first = a[i]; }
    else if (a[i] > second && a[i] != first) second = a[i];
}
if (second == INT_MIN) printf("No second largest\n");
else printf("Second largest = %d\n", second);
```

{12, 35, 1, 10, 34, 1} → 34 · O(n). Finding it by sorting takes O(n log n) — unnecessary.

P26. Reverse an array (in-place)

```
for (int i = 0, j = n - 1; i < j; i++, j--) {
    int t = a[i]; a[i] = a[j]; a[j] = t;
}
```

{1, 2, 3, 4, 5} → {5, 4, 3, 2, 1} · O(n), Space O(1)

P27. Linear and Binary search BSCS AP 17.01.2026

```
int linearSearch(const int a[], int n, int key){
    for (int i = 0; i < n; i++)
        if (a[i] == key) return i;
    return -1;                                  /* not found */
}

int binarySearch(const int a[], int n, int key){   /* a must be sorted */
    int lo = 0, hi = n - 1;
    while (lo <= hi) {
        int mid = lo + (hi - lo) / 2;           /* avoids the overflow of (lo+hi)/2 */
        if (a[mid] == key) return mid;
        else if (a[mid] < key) lo = mid + 1;    /* search the right half */
        else hi = mid - 1;                      /* search the left half */
    }
    return -1;
}
```

Step (key = 23)	lo	hi	mid	a[mid]	Decision
1	0	9	4	16	16 < 23 → lo = 5
2	5	9	7	56	56 > 23 → hi = 6
3	5	6	5	23	Found → index 5

Array {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}. Binary O(log n), Linear O(n). Binary search fails on an unsorted array, because the decision "if a[mid] < key, the key is on the right" is not true then.

P28. Bubble sort (with early-exit flag)

```
void bubbleSort(int a[], int n){
    for (int i = 0; i < n - 1; i++) {
        int swapped = 0;
        for (int j = 0; j < n - 1 - i; j++)        /* the last i items are already sorted
*/
            if (a[j] > a[j + 1]) {
                int t = a[j]; a[j] = a[j + 1]; a[j + 1] = t;
                swapped = 1;
            }
        if (!swapped) break;                        /* no swap means sorted → best case
O(n) */
    }
}
```

P29. Selection sort

```
void selectionSort(int a[], int n){
    for (int i = 0; i < n - 1; i++) {
        int m = i;                                  /* index of the smallest in the
remaining part */
        for (int j = i + 1; j < n; j++)
            if (a[j] < a[m]) m = j;
        if (m != i) { int t = a[i]; a[i] = a[m]; a[m] = t; }
    }
}
```

P30. Insertion sort — full program BSCS Officer IT 2020

```c
#include <stdio.h>

void insertionSort(int a[], int n){
    for (int i = 1; i < n; i++) {
        int key = a[i], j = i - 1;
        while (j >= 0 && a[j] > key) {     /* shift larger items one place right */
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;                      /* place the key in its correct position */
    }
}

int main(void){
    int n, a[100];
    printf("How many numbers (<= 100)? ");
    scanf("%d", &n);
    if (n < 1 || n > 100) { printf("Invalid n\n"); return 1; }
    for (int i = 0; i < n; i++) scanf("%d", &a[i]);
    insertionSort(a, n);
    for (int i = 0; i < n; i++) printf("%d ", a[i]);
    printf("\n");
    return 0;
}
```

Input: 6 / 5 2 4 6 1 3 → 1 2 3 4 5 6 · The state after each pass is in the sorting table in Chapter 4.

Sort	Best	Average	Worst	Stable	When it is good
Bubble (flag)	O(n)	O(n^2)	O(n^2)	Yes	For teaching; small, nearly sorted data
Selection	O(n^2)	O(n^2)	O(n^2)	No	Fewest swaps (≤ n − 1)
Insertion	O(n)	O(n^2)	O(n^2)	Yes	Small or nearly sorted data, online input
Merge	O(n log n)	O(n log n)	O(n log n)	Yes	Large data, linked lists; extra O(n) memory

Why Bubble is inefficient on large data and why Merge is better BSCS AP 17.01.2026

In each pass, Bubble sort compares only adjacent pairs and sends one element to its correct place — about $n^2/2$ comparisons for n elements. For 10^6 elements, ≈ 5×10^{11} comparisons. Merge sort repeatedly halves the array ($\log_2 n$ levels) and merges with n comparisons per level — a total of n $\log_2 n \approx 2 \times 10^7$ comparisons, i.e. about 25,000 times fewer.

P31. Remove duplicates from a sorted array

```c
int k = 0;                                   /* k = new length */
for (int i = 0; i < n; i++)
    if (i == 0 || a[i] != a[i - 1]) a[k++] = a[i];
```

{1, 1, 2, 3, 3, 3, 4} → {1, 2, 3, 4}, k = 4 · O(n). If unsorted, sort first or use nested loops (O(n^2)).

P32. Frequency of each element

```
int a[] = {2, 3, 2, 5, 3, 2}, n = 6, done[6] = {0};
for (int i = 0; i < n; i++) {
    if (done[i]) continue;                  /* already counted */
    int c = 1;
    for (int j = i + 1; j < n; j++)
        if (a[j] == a[i]) { c++; done[j] = 1; }
    printf("%d -> %d\n", a[i], c);
}
```

Output: 2 -> 3, 3 -> 2, 5 -> 1 · $O(n^2)$. If the value range is small, use a count array for O(n).

P33. Matrix addition, multiplication and transpose

```
#include <stdio.h>
#define N 2
int main(void){
    int A[N][N] = {{1, 2}, {3, 4}}, B[N][N] = {{5, 6}, {7, 8}};
    int S[N][N], P[N][N], T[N][N];
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            S[i][j] = A[i][j] + B[i][j];        /* addition */
            T[j][i] = A[i][j];                  /* transpose */
            P[i][j] = 0;                        /* multiplication: row × column */
            for (int k = 0; k < N; k++) P[i][j] += A[i][k] * B[k][j];
        }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) printf("%4d", P[i][j]);
        printf("\n");
    }
    return 0;
}
```

Sum = [[6, 8], [10, 12]] · Product = [[19, 22], [43, 50]] (first row: 1·5 + 2·7 = 19, 1·6 + 2·8 = 22) · Transpose(A) = [[1, 3], [2, 4]]. Condition for multiplication: columns of A = rows of B; time $O(n^3)$.

3.6 Strings

P34. Reverse a string and palindrome

```
#include <stdio.h>
#include <string.h>
int main(void){
    char s[101];
    scanf("%100s", s);                      /* one word; use fgets for a line */
    int n = strlen(s), isPal = 1;
    for (int i = 0, j = n - 1; i < j; i++, j--)
        if (s[i] != s[j]) { isPal = 0; break; }
    printf("%s\n", isPal ? "Palindrome" : "Not palindrome");
    for (int i = 0, j = n - 1; i < j; i++, j--) { char t = s[i]; s[i] = s[j]; s[j] = t; }
    printf("Reversed: %s\n", s);
    return 0;
}
```

```
// Java:   String r = new StringBuilder(s).reverse().toString();  boolean pal =
s.equals(r);
# Python: print(s[::-1], s == s[::-1])
```

madam → Palindrome, Reversed: madam · hello → Not palindrome, Reversed: olleh · O(n)

P35. Count vowels, consonants and words

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
int main(void){
    char s[201];
    fgets(s, sizeof s, stdin);               /* whole line, including spaces */
    int v = 0, c = 0, words = 0, inWord = 0;
    for (int i = 0; s[i] != '\0'; i++) {
        char ch = tolower((unsigned char) s[i]);
        if (isalpha((unsigned char) ch)) {
            if (strchr("aeiou", ch)) v++; else c++;
        }
        if (isspace((unsigned char) s[i])) inWord = 0;
        else if (!inWord) { inWord = 1; words++; }   /* a new word starts */
    }
    printf("Vowels=%d Consonants=%d Words=%d\n", v, c, words);
    return 0;
}
```

Input: Bangladesh Bank → Vowels=4 Consonants=10 Words=2. The word count stays correct even with multiple spaces.

P36. Anagram check (listen ↔ silent)

```
int isAnagram(const char *a, const char *b){   /* assume only lowercase a-z */
    int cnt[26] = {0};
    if (strlen(a) != strlen(b)) return 0;
    for (int i = 0; a[i] != '\0'; i++) { cnt[a[i] - 'a']++; cnt[b[i] - 'a']--; }
    for (int i = 0; i < 26; i++) if (cnt[i] != 0) return 0;
    return 1;
}
```

("listen", "silent") → 1 · ("bank", "bake") → 0 · O(n). Python: `sorted(a) == sorted(b)`

3.7 Recursion

The two parts of writing recursion

Base case (where to stop) + **Recursive case** (the function calls itself on a smaller problem). Without a base case, stack overflow occurs. Each call keeps one frame on the stack, so space is O(depth).

P37. Recursive factorial — algorithm and code BSCS Officer IT 2020

```
Algorithm FACT(n)
1. If n = 0 or n = 1, return 1          // base case
2. Else return n × FACT(n - 1)          // recursive case
```

```
unsigned long long fact(int n){
    if (n <= 1) return 1;
    return n * fact(n - 1);
}
```

fact(4) = 4 × fact(3) = 4 × 3 × fact(2) = 4 × 3 × 2 × fact(1) = 24 · Time O(n), Space O(n) (stack)

P38. Recursive Fibonacci

```
int fib(int n){ return n <= 1 ? n : fib(n - 1) + fib(n - 2); }
```

fib(5) = 5; total number of calls is 15. Time $O(2^n)$ — the same values are computed again and again; with memoization or a loop (P5) it is O(n).

P39. Recursive power and digit sum

```
long long powr(long long x, int n){           /* O(log n): xⁿ = (x^(n/2))² */
    if (n == 0) return 1;
    long long h = powr(x, n / 2);
    return (n % 2 == 0) ? h * h : h * h * x;
}
int digitSum(int n){ return n == 0 ? 0 : n % 10 + digitSum(n / 10); }
```

powr(3, 5) = 243 · digitSum(1234) = 10

P40. Tower of Hanoi

```
void hanoi(int n, char from, char to, char via){
    if (n == 0) return;
    hanoi(n - 1, from, via, to);              /* top n-1 disks to the helper peg */
    printf("Move disk %d from %c to %c\n", n, from, to);
    hanoi(n - 1, via, to, from);              /* those disks to the target */
}
/* hanoi(3, 'A', 'C', 'B'); */
```

```
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

Total moves = $2^n - 1$ (n = 3 → 7) · Time $O(2^n)$

3.8 Java OOP Programs

P41. Class, constructor overloading and this()

```
class Student {
    private String name;
    private int roll;

    Student() { this("Unknown", 0); }                /* constructor chaining */
    Student(String name, int roll) { this.name = name; this.roll = roll; }

    void show() { System.out.println(roll + " " + name); }
}
public class Main {
    public static void main(String[] args) {
        new Student("Rahim", 7).show();
        new Student().show();
    }
}
```

Output: 7 Rahim / 0 Unknown. A constructor has no return type and has the same name as the class; private field = encapsulation.

P42. Inheritance + method overriding + runtime polymorphism Officer IT 2026 (25105)

```
class Shape {
    double area() { return 0; }
    String name() { return "Shape"; }
}
class Circle extends Shape {
    private double r;
    Circle(double r) { this.r = r; }
    @Override double area() { return Math.PI * r * r; }
    @Override String name() { return "Circle"; }
}
class Rectangle extends Shape {
    private double w, h;
    Rectangle(double w, double h) { this.w = w; this.h = h; }
    @Override double area() { return w * h; }
    @Override String name() { return "Rectangle"; }
}
public class Main {
    public static void main(String[] args) {
        Shape[] shapes = { new Circle(1), new Rectangle(3, 4) };
        for (Shape s : shapes)                       /* reference is Shape, objects differ
*/
            System.out.printf("%s: %.2f%n", s.name(), s.area());
    }
}
```

Output: Circle: 3.14 / Rectangle: 12.00. Which `area()` runs is decided at runtime by looking at the object (dynamic dispatch).

P43. Interface

```
interface Payable { double getPayment(); }          /* contract: what must be done */

class Invoice implements Payable {
    private int qty; private double price;
    Invoice(int qty, double price) { this.qty = qty; this.price = price; }
    public double getPayment() { return qty * price; }
}
class SalariedEmployee implements Payable {
    private double salary;
    SalariedEmployee(double salary) { this.salary = salary; }
    public double getPayment() { return salary; }
}
public class Main {
    public static void main(String[] args) {
        Payable[] items = { new Invoice(3, 250.0), new SalariedEmployee(45000) };
        double total = 0;
        for (Payable p : items) total += p.getPayment();
        System.out.println("Total = " + total);
    }
}
```

Output: Total = 45750.0. A class can implement multiple interfaces (Java's alternative to multiple inheritance).

P44. Abstract class

```
abstract class Account {
    protected double balance;
    Account(double balance) { this.balance = balance; }
    abstract double interestRate();                  /* the subclass must provide it */
    double yearlyInterest() { return balance * interestRate(); }   /* concrete method */
}
class Savings extends Account {
    Savings(double b) { super(b); }
    double interestRate() { return 0.05; }
}
class Current extends Account {
    Current(double b) { super(b); }
    double interestRate() { return 0.0; }
}
public class Main {
    public static void main(String[] args) {
        Account a = new Savings(10000), b = new Current(10000);
        System.out.println(a.yearlyInterest() + " " + b.yearlyInterest());
    }
}
```

Output: 500.0 0.0. An object of an abstract class cannot be created; it can have a constructor, fields and concrete methods — this is the main difference from an interface.

P45. Exception handling — try / catch / finally

```
public class Demo {
    public static void main(String[] args) {
        int[] a = {10, 0};
        try {
            System.out.println(a[0] / a[1]);
        } catch (ArithmeticException e) {
            System.out.println("Divide by zero");
        } finally {
            System.out.println("finally runs");           /* always runs */
        }
        try {
            System.out.println(a[2]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Bad index");
        }
    }
}
```

Output: Divide by zero / finally runs / Bad index

Topic	Checked exception	Unchecked exception
Examples	IOException, SQLException	ArithmeticException, NullPointerException
When it is caught	Must be handled/declared at compile time	At runtime; handling is not mandatory
Parent	Exception (except RuntimeException)	RuntimeException

`throw` = throws an exception (inside a method); `throws` = a declaration in the method signature that it may throw an exception.

P46. BankAccount — deposit, withdraw and custom exception

```
class InsufficientBalanceException extends Exception {     /* checked */
    public InsufficientBalanceException(String msg) { super(msg); }
}

class BankAccount {
    private final String accNo;
    private double balance;                                  /* cannot be changed directly
from outside */

    public BankAccount(String accNo, double opening) {
        this.accNo = accNo;
        this.balance = opening;
    }
    public void deposit(double amt) {
        if (amt <= 0) throw new IllegalArgumentException("Invalid amount");
        balance += amt;
    }
    public void withdraw(double amt) throws InsufficientBalanceException {
        if (amt <= 0) throw new IllegalArgumentException("Invalid amount");
        if (amt > balance)
            throw new InsufficientBalanceException("Balance " + balance + " < " + amt);
        balance -= amt;
    }
    public double getBalance() { return balance; }
    public String getAccNo()   { return accNo; }
}

public class Main {
    public static void main(String[] args) {
        BankAccount a = new BankAccount("1001", 5000);
        a.deposit(2000);
        try {
            a.withdraw(3000);
            System.out.println("Balance: " + a.getBalance());
            a.withdraw(10000);
        } catch (InsufficientBalanceException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            System.out.println("Final: " + a.getBalance());
        }
    }
}
```

```
Balance: 4000.0
Error: Balance 4000.0 < 10000.0
Final: 4000.0
```

One program shows encapsulation (private balance + getter), validation, a checked custom exception and finally — directly useful for bank exam questions.

3.9 Output Tracing — 21 Problems

> **Technique**
> Keep one column on paper for each variable and write its value after every line. In recursion, every call has its own separate local variables — draw the call tree.

T1. Post- and pre-increment

```
int i = 5;
printf("%d ", i++);
printf("%d ", ++i);
printf("%d", i);
```

Output: `5 7 7` — i++ gives the value first (5), then increments (6); ++i increments first (7), then gives the value.

T2. Integer division and cast

```
int a = 7, b = 2;
float x = a / b;
float y = (float) a / b;
printf("%d %.1f %.1f %d\n", a / b, x, y, a % b);
printf("%d %d", -7 / 2, -7 % 2);
```

Output:
`3 3.0 3.5 1`
`-3 -1` — int/int = int (fraction dropped); even though it is stored in a float, it already became 3 earlier; in C99, division truncates toward zero, and the remainder has the sign of the dividend.

T3. Char arithmetic

```
char c = 'A';
c = c + 2;
printf("%c %d %c", c, c, c + 32);
```

Output: `C 67 c` — 'A' = 65, +2 = 67 = 'C'; a lowercase letter is 32 more (99 = 'c').

T4. Static local variable

```
void f(void){ static int s = 0; int a = 0; s++; a++; printf("%d %d\n", s, a); }
int main(void){ f(); f(); f(); return 0; }
```

Output: `1 1` / `2 1` / `3 1` — a static variable is initialized only once and keeps its value between calls; an ordinary local variable is new every time.

T5. Recursion — print before and after AP 2024 (10198)

```
#include <stdio.h>
void fun(int i){
    if (i <= 0) return;
    printf("%d, ", i--);
    fun(--i);
    printf("%d, ", i);
}
int main(void){ fun(5); return 0; }
```

Call	First printf	Goes to next call	i after return	Second printf
fun(5)	5 (i → 4)	--i = 3 → fun(3)	3	3
fun(3)	3 (i → 2)	--i = 1 → fun(1)	1	1
fun(1)	1 (i → 0)	--i = −1 → fun(−1) — base case	−1	−1

Output: `5, 3, 1, -1, 1, 3,` — the second printf calls print in reverse order (the inner call finishes first).

> **Typos in the question paper**
>
> The original question printed `int fun`, `if(I<=Ø)` and `Printf`. C is case-sensitive, so writing it exactly would give a compile error — begin your answer with "Assuming I = i, Printf = printf, Ø = 0". A valueless `return;` in an `int` function gives a warning in C; the output is the same.

T6. Pointer and array

```
int a[] = {10, 20, 30, 40};
int *p = a;
printf("%d ", *p + 1);
printf("%d ", *(p + 1));
p += 2;
printf("%d %d ", *p, p[-1]);
printf("%d", *++p);
```

Output: `11 20 30 20 40` — *p + 1 = value + 1; *(p+1) = next element; p[-1] = *(p−1); *++p increments the pointer first.

T7. Call by value vs call by reference

```
void s1(int x, int y){ int t = x; x = y; y = t; }
void s2(int *x, int *y){ int t = *x; *x = *y; *y = t; }
int main(void){
    int a = 1, b = 2;
    s1(a, b);   printf("%d %d ", a, b);
    s2(&a, &b); printf("%d %d", a, b);
    return 0;
}
```

Output: `1 2 2 1` — s1 changes copies; s2 changes the original values through addresses.

T8. Switch fall-through

```
int x = 2;
switch (x) {
    case 1: printf("one ");
    case 2: printf("two ");
    case 3: printf("three ");
            break;
    default: printf("other");
}
```

Output: `two three` — case 2 has no break, so the cases below it also run until a break.

T9. Short-circuit evaluation

```
int i = 0, j = 5;
if (i && ++j) ;
printf("%d ", j);
if (i || ++j) ;
printf("%d", j);
```

Output: `5 6` — if the left side of && is false, the right side does not run; the left side of || is false, so the right side (++j) runs.

T10. Ternary operator

```
int a = 10, b = 20, x = 5;
int m = (a > b) ? a : b;
printf("%d %d %d", m, (a == 10) ? (b > 15 ? 1 : 2) : 3, (x < 0) ? 10 : 15);
```

Output: `20 1 15`

T11. break and continue

```
for (int i = 1; i <= 10; i++) {
    if (i % 2 == 0) continue;
    if (i > 7) break;
    printf("%d ", i);
}
```

Output: `1 3 5 7` — even numbers are skipped; break at i = 9.

T12. while (n--)

```
int n = 3;
while (n--) printf("%d ", n);
printf("| %d", n);
```

Output: `2 1 0 | -1` — the condition tests the old value, then decrements; at the last test it sees 0 (false) and n becomes −1.

T13. Recursion — print before and after the call

```
void p(int n){
    if (n == 0) return;
    printf("%d ", n);
    p(n - 1);
    printf("%d ", n);
}
/* p(3); */
```

Output: `3 2 1 1 2 3`

T14. Nested loop count

```
int count = 0;
for (int i = 0; i < 4; i++)
    for (int j = i; j < 4; j++) count++;
printf("%d", count);
```

Output: `10` — 4 + 3 + 2 + 1 = n(n+1)/2.

T15. Java: String == vs equals()

```
String a = "bank";
String b = "bank";
String c = new String("bank");
System.out.println(a == b);
System.out.println(a == c);
System.out.println(a.equals(c));
```

Output: `true` / `false` / `true` — literals are the same String pool object; `new` creates a new object; == compares references, equals() compares content.

T16. Java: overloading resolution

```
static void f(int x)     { System.out.println("int"); }
static void f(long x)    { System.out.println("long"); }
static void f(double x)  { System.out.println("double"); }
static void f(Integer x) { System.out.println("Integer"); }

byte b = 5;
f(b); f(5L); f(5.0f); f('A');
```

Output: `int` / `long` / `double` / `int` — the compiler first looks for the nearest widening (byte→int, float→double, char→int); boxing (Integer) is used only if that fails.

T17. Java: static vs dynamic binding

```
class A {
    static void s() { System.out.println("A.s"); }
    void m()        { System.out.println("A.m"); }
}
class B extends A {
    static void s() { System.out.println("B.s"); }      /* hiding */
    @Override void m() { System.out.println("B.m"); }   /* overriding */
}
A obj = new B();
obj.s();
obj.m();
```

Output: `A.s` / `B.m` — a static method is decided at compile time by the reference type (A); an instance method is decided at runtime by the object (B).

T18. Java: field hiding vs method overriding

```
class P { int x = 10; int get() { return x; } }
class C extends P { int x = 20; @Override int get() { return x; } }
P p = new C();
System.out.println(p.x + " " + p.get());
```

Output: `10 20` — fields have no polymorphism (x of reference type P); the method is overridden.

T19. Java: constructor execution order

```
class A { A() { System.out.print("A "); } }
class B extends A { B() { System.out.print("B "); } }
class C extends B { C() { super(); System.out.print("C"); } }
new C();
```

Output: `A B C` — every constructor first calls super() (even if invisible), so the parent runs first.

T20. Java: compound assignment and increment

```
int x = 5;
x += x++ + ++x;
System.out.println(x);
```

Output: `17` — in Java, evaluation is left to right: old x = 5 is saved; x++ gives 5 (x = 6); ++x gives 7; 5 + (5 + 7) = 17.

> **Same code in C = Undefined Behaviour**
>
> In C, changing a variable more than once in the same expression (without a sequence point) gives compiler-dependent results — if such a question appears in the exam, write "undefined behaviour" and give the reason.

T21. Java: String concatenation

```
System.out.println(1 + 2 + "3" + 4 + 5);
System.out.println("" + 'a' + 'b');
System.out.println('a' + 'b' + "");
```

Output: `3345` / `ab` / `195` — left to right; before a String appears, numbers are added (1+2 = 3), after that everything is concatenated; char + char = int (97 + 98).

Chapter 4 — SQL and Data Structures: Hands-On

Every AP written exam has had at least one SQL question (romantic comedy, S-R-Q join, Department/Employee, IT department staff) and one "show by hand" DSA question (max-heap insert/extract, hashing with linear probing, adjacency list/matrix, LRU page fault, queue using two stacks). Part 1 of this chapter has 31 SQL queries (with results on sample data). Part 2 has step-by-step DSA tables.

4.1 Sample Schema and Data

All queries are written in MySQL 8 / standard SQL. Results of the first 10 queries are shown on the data below — verify them yourself.

DEPARTMENT

dept_id	dept_name	location
10	IT	Dhaka
20	Finance	Chattogram
30	HR	Dhaka
40	Audit	Khulna

EMPLOYEE

emp_id	name	dept_id	salary	commission	manager_id
1	Rahim	10	80000	NULL	NULL
2	Karim	10	60000	5000	1
3	Salma	20	70000	NULL	1
4	Nadia	20	50000	3000	3
5	Tanvir	10	60000	NULL	2
6	Jamil	30	40000	2000	1
7	Rupa	NULL	45000	NULL	1

Total salary 405000, average 57857.14. The Audit department has no employees; Rupa's department is NULL. Both are deliberate, to show the LEFT JOIN and NULL traps.

BRANCH (branch_id, branch_name, city)

id	name	city
1	Motijheel	Dhaka
2	Agrabad	Chattogram
3	Gulshan	Dhaka

CUSTOMER (cust_id, name, city)

id	name	city
101	Arif	Dhaka
102	Bina	Khulna
103	Chayan	Dhaka
104	Dola	Sylhet

ACCOUNT (acc_no, cust_id, branch_id, type, balance)

acc_no	cust	branch	type	balance
5001	101	1	SAV	150000
5002	101	3	CUR	20000
5003	102	2	SAV	80000
5004	103	1	SAV	5000
5005	103	2	CUR	300000

TRANSACTIONS (txn_id, acc_no, txn_type, amount, txn_date)

id	acc	type	amount	date
1	5001	D	50000	2026-09-02
2	5001	W	20000	2026-09-15

3	5003	D	10000	2026-09-20
4	5005	W	100000	2026-10-01
5	5001	D	5000	2026-10-05

4.2 SQL Query Bank (31 queries)

Q1. Condition and sorting — salary above 55,000, highest first

```
SELECT name, salary
FROM employee
WHERE salary > 55000
ORDER BY salary DESC, name ASC;     -- name order for equal salaries, so the result is
fixed
```

name	salary
Rahim	80000
Salma	70000
Karim	60000
Tanvir	60000

Q2. LIKE — pattern matching

```
SELECT name FROM employee WHERE name LIKE 'R%';    -- starts with R
SELECT name FROM employee WHERE name LIKE '%a';    -- ends with a
-- '_a%' = second letter is a;  % = any number of characters, _ = exactly one character
```

Result: **'R%'** → Rahim, Rupa · **'%a'** → Salma, Nadia, Rupa · **'_a%'** → all 6 except Rupa.

Q3. IN and BETWEEN

```
SELECT name, dept_id, salary
FROM employee
WHERE dept_id IN (20, 30)
  AND salary BETWEEN 40000 AND 60000;   -- BETWEEN includes both ends (inclusive)
```

name	dept_id	salary
Nadia	20	50000
Jamil	30	40000

Q4. NULL handling — total income = salary + commission

```
SELECT name, salary + COALESCE(commission, 0) AS total_income   -- MySQL:
IFNULL(commission, 0)
FROM employee;

SELECT name FROM employee WHERE commission IS NULL;     -- "= NULL" never works
```

name	total_income
Rahim	80000
Karim	65000
Salma	70000
Nadia	53000
Tanvir	60000
Jamil	42000
Rupa	45000

commission IS NULL → Rahim, Salma, Tanvir, Rupa.

> **The NULL trap**
>
> If you write `salary + commission`, the total income of anyone with a NULL commission also becomes NULL. Any arithmetic with NULL gives NULL. Aggregates (SUM, AVG, COUNT(col)) skip NULLs, but COUNT(*) counts all rows.

Q5. Aggregate functions

```
SELECT COUNT(*) AS total, COUNT(commission) AS with_comm,
       SUM(salary) AS total_sal, ROUND(AVG(salary), 2) AS avg_sal,
       MAX(salary) AS max_sal, MIN(salary) AS min_sal
FROM employee;
```

total	with_comm	total_sal	avg_sal	max_sal	min_sal
7	3	405000	57857.14	80000	40000

Q6. GROUP BY + HAVING — employees and total salary per department (at least 2 employees)

```
SELECT d.dept_name, COUNT(*) AS emp_count,
       SUM(e.salary) AS total_salary, ROUND(AVG(e.salary), 2) AS avg_salary
FROM employee e
JOIN department d ON e.dept_id = d.dept_id
GROUP BY d.dept_name
HAVING COUNT(*) >= 2                -- condition on a group: HAVING; on a row: WHERE
ORDER BY total_salary DESC;
```

dept_name	emp_count	total_salary	avg_salary
IT	3	200000	66666.67
Finance	2	120000	60000.00

Written order vs execution order

Written: SELECT → FROM → WHERE → GROUP BY → HAVING → ORDER BY → LIMIT. Executed: **FROM/JOIN → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT**. So an aggregate (COUNT, AVG) cannot be used in WHERE, and a SELECT alias cannot be used in WHERE (it can be used in ORDER BY).

Q7. INNER JOIN — employee name, department and location

```
SELECT e.name, d.dept_name, d.location
FROM employee e
INNER JOIN department d ON e.dept_id = d.dept_id
ORDER BY e.emp_id;
```

name	dept_name	location
Rahim	IT	Dhaka
Karim	IT	Dhaka
Salma	Finance	Chattogram
Nadia	Finance	Chattogram
Tanvir	IT	Dhaka
Jamil	HR	Dhaka

Rupa is missing (NULL dept_id matches no row). Audit is missing too (no employees).

Q8. LEFT JOIN — employee count of every department (including zero)

```
SELECT d.dept_name, COUNT(e.emp_id) AS emp_count    -- COUNT(*) would give 1 for Audit!
FROM department d
LEFT JOIN employee e ON d.dept_id = e.dept_id
GROUP BY d.dept_id, d.dept_name;
```

dept_name	emp_count
IT	3
Finance	2
HR	1
Audit	0

Q9. SELF JOIN — employee and their manager

```
SELECT e.name AS employee, m.name AS manager
FROM employee e
JOIN employee m ON e.manager_id = m.emp_id;     -- same table under two aliases
```

employee	manager
Karim	Rahim

Salma	Rahim
Nadia	Salma
Tanvir	Karim
Jamil	Rahim
Rupa	Rahim

With LEFT JOIN, the row Rahim → NULL also appears (he has no manager).

Q10. Salary above the average (subquery)

```
SELECT name, salary
FROM employee
WHERE salary > (SELECT AVG(salary) FROM employee);     -- average 57857.14
```

name	salary
Rahim	80000
Karim	60000
Salma	70000
Tanvir	60000

Q11. Above own department's average (correlated subquery)

```
SELECT e1.name, e1.dept_id, e1.salary
FROM employee e1
WHERE e1.salary > (SELECT AVG(e2.salary)
                   FROM employee e2
                   WHERE e2.dept_id = e1.dept_id);   -- a separate average for each outer
row
```

Department averages: IT 66666.67, Finance 60000, HR 40000 → result: **Rahim (80000), Salma (70000)**. Jamil's salary equals the average (not above it). Rupa's dept is NULL, so the subquery average is NULL → comparison is unknown → excluded.

Q12. N-th highest salary — four methods

```
-- (a) Second highest: MAX among those smaller than MAX
SELECT MAX(salary) FROM employee
WHERE salary < (SELECT MAX(salary) FROM employee);              -- 70000

-- (b) LIMIT/OFFSET: N-th = OFFSET N-1 (DISTINCT is essential)
SELECT DISTINCT salary FROM employee
ORDER BY salary DESC LIMIT 1 OFFSET 1;                           -- 70000

-- (c) Window function: all employees with the third highest salary
SELECT name, salary FROM (
    SELECT name, salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS rk
    FROM employee) t
WHERE rk = 3;                                                    -- Karim, Tanvir (60000)

-- (d) Correlated: for the N-th, exactly N-1 distinct salaries are greater
SELECT DISTINCT salary FROM employee e1
WHERE 2 = (SELECT COUNT(DISTINCT e2.salary) FROM employee e2
           WHERE e2.salary > e1.salary);                         -- N = 3 → 60000
```

salary (DESC)	80000	70000	60000	60000	50000	45000	40000
ROW_NUMBER	1	2	3	4	5	6	7
RANK	1	2	3	3	5	6	7
DENSE_RANK	1	2	3	3	4	5	6

Q13. Finding and deleting duplicates

```
SELECT salary, COUNT(*) AS cnt
FROM employee
GROUP BY salary
HAVING COUNT(*) > 1;                          -- 60000 → 2

-- delete duplicate rows with the same email, keeping the smallest id (MySQL)
DELETE c1 FROM customer_email c1
JOIN customer_email c2
  ON c1.email = c2.email AND c1.id > c2.id;
```

Q14. EXISTS / NOT EXISTS — and the NOT IN NULL trap

```
-- departments with no employees
SELECT d.dept_name FROM department d
WHERE NOT EXISTS (SELECT 1 FROM employee e WHERE e.dept_id = d.dept_id);   -- Audit

-- same task with NOT IN — but the result is empty!
SELECT dept_name FROM department
WHERE dept_id NOT IN (SELECT dept_id FROM employee);                       -- no rows

-- fix
SELECT dept_name FROM department
WHERE dept_id NOT IN (SELECT dept_id FROM employee WHERE dept_id IS NOT NULL);  -- Audit
```

Why NOT IN failed

The subquery contains Rupa's NULL. `40 NOT IN (10, 20, 10, …, NULL)` means `40 <> 10 AND … AND 40 <> NULL`. The last part is UNKNOWN, so the whole condition is never TRUE. NOT EXISTS has no such problem — use NOT EXISTS on nullable columns.

Q15. Total balance per customer (more than 1 lakh)

```
SELECT c.name, SUM(a.balance) AS total_balance
FROM customer c
JOIN account a ON a.cust_id = c.cust_id
GROUP BY c.cust_id, c.name
HAVING SUM(a.balance) > 100000;
```

Result: Arif 170000, Chayan 305000 (Bina 80000 excluded).

Q16. Number of accounts and total deposits per branch

```
SELECT b.branch_name, COUNT(a.acc_no) AS accounts, COALESCE(SUM(a.balance), 0) AS total
FROM branch b
LEFT JOIN account a ON a.branch_id = b.branch_id
GROUP BY b.branch_id, b.branch_name
ORDER BY total DESC;
```

Result: Agrabad 2 accounts, 380000 · Motijheel 2 accounts, 155000 · Gulshan 1 account, 20000.

Q17. Customers with no account

```
SELECT c.name
FROM customer c
LEFT JOIN account a ON a.cust_id = c.cust_id
WHERE a.acc_no IS NULL;           -- anti-join
```

Result: Dola.

Q18. Accounts with no transaction

```
SELECT a.acc_no
FROM account a
WHERE NOT EXISTS (SELECT 1 FROM transactions t WHERE t.acc_no = a.acc_no);
```

Result: 5002, 5004.

Q19. Monthly deposits and withdrawals (pivot using CASE)

```
SELECT DATE_FORMAT(txn_date, '%Y-%m') AS month,
       SUM(CASE WHEN txn_type = 'D' THEN amount ELSE 0 END) AS deposits,
       SUM(CASE WHEN txn_type = 'W' THEN amount ELSE 0 END) AS withdrawals
FROM transactions
GROUP BY DATE_FORMAT(txn_date, '%Y-%m')
ORDER BY month;
```

month	deposits	withdrawals

2026-09	60000	20000
2026-10	5000	100000

Q20. Many-to-many: movies that are both Romance *and* Comedy BB AP 2023

Schema: MOVIE(movie_id, title), GENRE(genre_id, genre), HAS_GENRE(movie_id, genre_id) — a junction table. Sample: Movie A → Romance, Comedy; Movie B → Romance; Movie C → Comedy, Action.

```
-- Method 1: join the junction table twice (once for Romance, once for Comedy)
SELECT COUNT(DISTINCT m.movie_id) AS romcom
FROM movie m
JOIN has_genre hg1 ON hg1.movie_id = m.movie_id
JOIN genre g1      ON g1.genre_id = hg1.genre_id AND g1.genre = 'Romance'
JOIN has_genre hg2 ON hg2.movie_id = m.movie_id
JOIN genre g2      ON g2.genre_id = hg2.genre_id AND g2.genre = 'Comedy';

-- Method 2: GROUP BY + HAVING COUNT(DISTINCT)
SELECT COUNT(*) AS romcom FROM (
    SELECT hg.movie_id
    FROM has_genre hg
    JOIN genre g ON g.genre_id = hg.genre_id
    WHERE g.genre IN ('Romance', 'Comedy')
    GROUP BY hg.movie_id
    HAVING COUNT(DISTINCT g.genre) = 2
) t;
```

Result: **1** (only Movie A).

> **The most common mistake**
>
> `WHERE genre = 'Romance' AND genre = 'Comedy'` — one row cannot have both genres at once, so the result is always 0. With `OR`, a movie is counted if it has either one (result 3). Both are wrong.

Q21. S–R–Q–U: related (sid, uid) pairs Combined 5 Bank AP 2023

S(sid, A), T(tid, B), U(uid, C), R(sid, tid, D) — many-to-many between S and T; Q(tid, uid, E) — many-to-many between T and U. Return (sid, uid) without SELECT DISTINCT.

```
SELECT R.sid, Q.uid
FROM R
JOIN Q ON R.tid = Q.tid;          -- related through the same T record

-- to make sure the records exist in S and U (same result if FKs exist)
SELECT S.sid, U.uid
FROM S
JOIN R ON S.sid = R.sid
JOIN Q ON R.tid = Q.tid
JOIN U ON Q.uid = U.uid;
```

If the same (sid, uid) pair is linked through more than one T, it appears more than once. The question forbids DISTINCT, so this is expected. State this in one line in your answer.

Q22. Names of employees in the IT department — logical steps BSCS AP 03.01.2026

```
-- Step 1: find the DEPARTMENTS row with DeptName = 'IT' (its Dept value)
-- Step 2: join EMPLOYEES on DeptID = Dept
-- Step 3: take only the Name column
SELECT e.Name
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DeptID = d.Dept
WHERE d.DeptName = 'IT';

-- subquery alternative
SELECT Name FROM EMPLOYEES
WHERE DeptID = (SELECT Dept FROM DEPARTMENTS WHERE DeptName = 'IT');
```

Q23. Department and Employee tables (a, b, c) AP 2024 (10198)

```
-- (a) total salary of each employee = salary + commission
SELECT EmpFname, Salary + COALESCE(Commission, 0) AS TotalSalary
FROM Employee;

-- (b) employees earning above the average — name and location, sorted by name
SELECT e.EmpFname, d.Location
FROM Employee e
JOIN Department d ON e.DeptCode = d.DeptCode
WHERE e.Salary > (SELECT AVG(Salary) FROM Employee)
ORDER BY e.EmpFname;

-- (c) employee count per name, smallest to largest count
SELECT EmpFname, COUNT(EmpCode) AS EmpCount
FROM Employee
GROUP BY EmpFname
ORDER BY EmpCount ASC;
```

Note: in the question's CREATE, Department.DeptCode is `int` but Employee.DeptCode is `varchar` — a type mismatch in the join. Writing "both should be INT, and Employee.DeptCode should be a FOREIGN KEY to Department" earns extra marks. The wording of (c) is unclear; write your own interpretation in one line.

Q24. CASE — salary grade

```
SELECT name, salary,
       CASE WHEN salary >= 70000 THEN 'High'
            WHEN salary >= 50000 THEN 'Medium'
            ELSE 'Low' END AS grade
FROM employee;
```

Rahim, Salma → High · Karim, Nadia, Tanvir → Medium · Jamil, Rupa → Low (CASE stops at the first true condition).

Q25. UPDATE and DELETE

```
UPDATE employee SET salary = salary * 1.05 WHERE dept_id = 10;     -- 5% raise in IT

UPDATE employee e
JOIN department d ON e.dept_id = d.dept_id
SET e.salary = e.salary + 2000
WHERE d.location = 'Dhaka';                                        -- MySQL join-update

DELETE FROM transactions WHERE txn_date < '2026-01-01';            -- without WHERE,
everything is deleted!
```

Q26. CREATE TABLE — PK, FK, CHECK, UNIQUE, DEFAULT

```
CREATE TABLE customer (
    cust_id   INT PRIMARY KEY,
    name      VARCHAR(100) NOT NULL,
    email     VARCHAR(100) UNIQUE,
    phone     VARCHAR(15)  UNIQUE,
    city      VARCHAR(50)  DEFAULT 'Dhaka'
);

CREATE TABLE account (
    acc_no     INT PRIMARY KEY,
    cust_id    INT NOT NULL,
    branch_id  INT NOT NULL,
    type       CHAR(3) NOT NULL CHECK (type IN ('SAV', 'CUR', 'FDR')),
    balance    DECIMAL(15, 2) NOT NULL DEFAULT 0 CHECK (balance >= 0),   -- negative
balance not allowed
    open_date  DATE DEFAULT (CURRENT_DATE),                               -- MySQL 8 needs
parentheses
    FOREIGN KEY (cust_id)   REFERENCES customer(cust_id),
    FOREIGN KEY (branch_id) REFERENCES branch(branch_id)
);

CREATE TABLE transactions (
    txn_id    INT AUTO_INCREMENT PRIMARY KEY,
    acc_no    INT NOT NULL,
    txn_type  CHAR(1) NOT NULL CHECK (txn_type IN ('D', 'W')),
    amount    DECIMAL(15, 2) NOT NULL CHECK (amount > 0),
    txn_date  DATE NOT NULL,
    FOREIGN KEY (acc_no) REFERENCES account(acc_no) ON DELETE RESTRICT
);
```

Use DECIMAL, not FLOAT, for money — to avoid floating-point rounding. CHECK is enforced from MySQL 8.0.16 onward.

Q27. ALTER TABLE

```
ALTER TABLE employee ADD COLUMN email VARCHAR(100) UNIQUE;
ALTER TABLE employee MODIFY COLUMN name VARCHAR(150) NOT NULL;     -- Oracle/SQL Server:
ALTER COLUMN
ALTER TABLE employee RENAME COLUMN email TO work_email;
ALTER TABLE employee DROP COLUMN work_email;
ALTER TABLE employee ADD CONSTRAINT fk_emp_dept
      FOREIGN KEY (dept_id) REFERENCES department(dept_id);
```

Q28. VIEW

```
CREATE VIEW dhaka_customer_balance AS
SELECT c.cust_id, c.name, SUM(a.balance) AS total_balance
FROM customer c JOIN account a ON a.cust_id = c.cust_id
WHERE c.city = 'Dhaka'
GROUP BY c.cust_id, c.name;

SELECT * FROM dhaka_customer_balance;          -- Arif 170000, Chayan 305000
```

A view is a stored query (virtual table). Benefits: it simplifies complex queries, and it adds security by hiding certain columns (a teller sees only the view, not the base table).

Q29. Transaction — fund transfer (Atomicity)

```
START TRANSACTION;

UPDATE account SET balance = balance - 10000
WHERE acc_no = 5001 AND balance >= 10000;     -- if affected rows is not 1 → ROLLBACK

UPDATE account SET balance = balance + 10000
WHERE acc_no = 5003;

INSERT INTO transactions (acc_no, txn_type, amount, txn_date)
VALUES (5001, 'W', 10000, CURRENT_DATE),
       (5003, 'D', 10000, CURRENT_DATE);

COMMIT;          -- if all succeed, made permanent;  if any step fails: ROLLBACK;
```

After: 5001 = 140000, 5003 = 90000. If power fails midway or the second UPDATE fails, ROLLBACK undoes the first one too — money is not "lost". `SAVEPOINT s1;` ... `ROLLBACK TO s1;` allows a partial rollback.

Q30. INDEX

```
CREATE INDEX idx_txn_acc_date ON transactions (acc_no, txn_date);
-- WHERE acc_no = 5001 AND txn_date BETWEEN ... — this query will use the index instead of
a full scan
DROP INDEX idx_txn_acc_date ON transactions;
```

An index (usually a B+ tree) reduces search from O(n) to about O(log n), but slows INSERT/UPDATE and uses space. In a composite index, searching by txn_date alone, without the leftmost column (acc_no), usually cannot use it.

Q31. GRANT / REVOKE (DCL)

```
CREATE USER 'teller'@'%' IDENTIFIED BY 'Str0ng#Pass';
GRANT SELECT, INSERT ON bankdb.transactions TO 'teller'@'%';
GRANT SELECT ON bankdb.dhaka_customer_balance TO 'teller'@'%';    -- view only
REVOKE INSERT ON bankdb.transactions FROM 'teller'@'%';
```

Least privilege: each user gets only the permissions needed for the job.

4.3 Max-Heap: insert and extract-root

Combined 5 Bank AP 2023

Question: Insert 5, 6, 8, Extract, Insert 4, 11, Extract, Insert 7, Extract, Extract — show the heap after each operation and state the value returned by each extract.

Rules (array-based, from index 0)

parent(i) = (i − 1)/2, left = 2i + 1, right = 2i + 2.
Insert: place at the end → if larger than its parent, swap upward (sift-up).
Extract: return the root → move the last element to the root → swap downward with the larger child (sift-down/heapify). Both are O(log n).

Step	Operation	What happens	Heap (array)	Returned
a	Insert 5	root of an empty heap	[5]	—
b	Insert 6	6 > 5 → swap	[6, 5]	—
c	Insert 8	at index 2, parent 6 < 8 → swap	[8, 5, 6]	—
d	Extract	remove 8, last element 6 to root; child 5 < 6 → OK	[6, 5]	**8**
e	Insert 4	parent 6 > 4 → OK	[6, 5, 4]	—
f	Insert 11	index 3, parent 5 → swap; parent 6 → swap	[11, 6, 4, 5]	—
g	Extract	remove 11, 5 to root; larger child 6 → swap	[6, 5, 4]	**11**
h	Insert 7	index 3, parent 5 → swap; parent 6 → swap	[7, 6, 4, 5]	—
i	Extract	remove 7, 5 to root; larger child 6 → swap	[6, 5, 4]	**7**
j	Extract	remove 6, 4 to root; child 5 > 4 → swap	[5, 4]	**6**

Order of values returned by extract: **8, 11, 7, 6**; final heap = [5, 4]. In the exam, draw a small tree beside each step.

4.4 Building a Min-Heap (bottom-up heapify)

Min-heap from the array [9, 4, 7, 1, 8, 2, 3]. Sift down from the last non-leaf index = ⌊n/2⌋ − 1 = 2 down to 0.

i	Element	Children	Action	Array
2	7	2, 3	swap with the smaller child, 2	[9, 4, 2, 1, 8, 7, 3]
1	4	1, 8	swap with 1	[9, 1, 2, 4, 8, 7, 3]

0	9	1, 2	swap with 1; then at index 1, 9's children are 4, 8 → swap with 4	[1, 4, 2, 9, 8, 7, 3]

Check: 1 ≤ (4, 2); 4 ≤ (9, 8); 2 ≤ (7, 3) ✔ · Bottom-up build is O(n); inserting one by one is O(n log n).

4.5 Hashing — the Same Key Set with Four Methods

Table size m = 11; keys in order: 10, 22, 31, 4, 15, 28, 17, 88, 59; h(k) = k mod 11 → 10, 0, 9, 4, 4, 6, 6, 0, 4.

(a) Linear probing: h(k, i) = (h(k) + i) mod 11

Key	h(k)	Probe sequence	Placed at
10, 22, 31, 4, 28	10, 0, 9, 4, 6	empty	10, 0, 9, 4, 6
15	4	4 ✗ → 5	5
17	6	6 ✗ → 7	7
88	0	0 ✗ → 1	1
59	4	4 ✗ → 5 ✗ → 6 ✗ → 7 ✗ → 8	8 (5 probes)

(b) Quadratic probing: h(k, i) = (h(k) + i^2) mod 11

10, 22, 31, 4, 28 go to the same places; 15: 4 ✗ → 4+1 = 5 · 17: 6 ✗ → 7 · 88: 0 ✗ → 1 · 59: 4 ✗ → 5 ✗ → 4+4 = 8 → **8 (3 probes)**. For this set the final table is the same as linear, but 59 needs fewer probes. Quadratic probing aims to avoid primary clustering (a tangle of consecutive filled slots).

(c) Double hashing: h(k, i) = (h_1(k) + i·h_2(k)) mod 11, h_2(k) = 1 + (k mod 10)

Key	h_1	h_2	Probe sequence	Placed at
15	4	6	4 ✗ → 10 ✗ → 16 mod 11 = 5	5
17	6	8	6 ✗ → 14 mod 11 = 3	3
88	0	9	0 ✗ → 9 ✗ → 18 mod 11 = 7	7
59	4	10	4 ✗ → 14 mod 11 = 3 ✗ → 24 mod 11 = 2	2

(d) Separate chaining (a linked list in each slot, new key added at the end)

Slot	0	1	2	3	4	5	6	7	8	9	10
Linear	22	88	—	—	4	15	28	17	59	31	10
Quadratic	22	88	—	—	4	15	28	17	59	31	10
Double	22	—	59	17	4	15	28	88	—	31	10
Chaining	22→88	—	—	—	4→15→59	—	28→17	—	—	31	10

Load factor α = n/m = 9/11 ≈ 0.82. In open addressing, α must be < 1, and probes grow quickly as α grows. Chaining works even with α > 1; average search is O(1 + α). In double hashing, h_2 must never be 0, and if m is prime, every slot is reachable.

4.6 Binary Search Tree — insert, traversal, delete

Insert order: 50, 30, 70, 20, 40, 60, 80, 35, 45, 65 (smaller goes left, larger goes right).

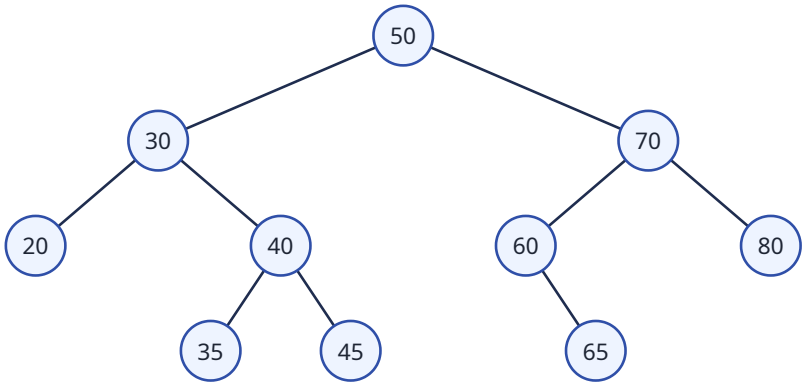


Figure 4.1: BST after inserting 50, 30, 70, 20, 40, 60, 80, 35, 45, 65

Traversal	Rule	Result
Inorder	Left → Root → Right	20 30 35 40 45 50 60 65 70 80 (always sorted)
Preorder	Root → Left → Right	50 30 20 40 35 45 70 60 65 80
Postorder	Left → Right → Root	20 35 45 40 30 65 60 80 70 50
Level order	level by level (queue)	50 30 70 20 40 60 80 35 45 65

Three cases of delete (each from the original tree separately)

Delete	Type	What to do	Result
20	Leaf	Remove it directly	30's left child becomes empty
60	One child (65)	Link the child to the parent	70's left child becomes 65
30	Two children	Replace the value with the inorder successor (smallest of the right subtree = 35), then delete 35	35 in place of 30; its children are 20 and 40; 40 has only a right child, 45

Search/insert/delete: O(log n) if balanced, but inserting in sorted order turns the tree into a linked list → O(n). This is why AVL/Red-Black trees exist.

4.7 AVL rotation

Balance factor (BF) = left height − right height. If |BF| > 1 at any node, rotate.

Case	Insert order	Problem	Fix	Result
LL	30, 20, 10	BF = +2 at 30, new node in left of left	right rotation at 30	20 (10, 30)
RR	10, 20, 30	BF = −2 at 10, right of right	left rotation at 10	20 (10, 30)
LR	30, 10, 20	BF = +2 at 30, right of left	left at 10, then right at 30	20 (10, 30)

RL	10, 30, 20	BF = −2 at 10, left of right	right at 30, then left at 10	20 (10, 30)

Example: insert 10, 20, 30, 40, 50, 25

10, 20, 30 → RR → 20(10, 30). 40 → right of 30. 50 → BF = −2 at 30 (RR) → left rotation at 30 → 20(10, 40(30, 50)).

25 → 20 → 40 → left of 30. Now BF at 20 = 1 − 3 = −2; BF of right child 40 = +1 → **RL**: right rotation at 40 (30 goes up), then left rotation at 20.

Final: root 30; left 20 (10, 25); right 40 (right child 50). |BF| ≤ 1 at every node ✔

4.8 Graph traversal: BFS and DFS

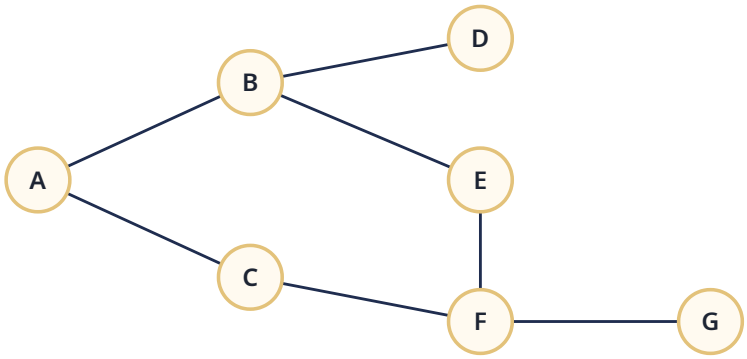


Figure 4.2: Undirected graph — A: B, C · B: A, D, E · C: A, F · D: B · E: B, F · F: C, E, G · G: F

BFS (from A, queue)

Dequeue	New enqueue	Queue
—	A	A
A	B, C	B C
B	D, E	C D E
C	F	D E F
D	—	E F
E	— (F already in)	F
F	G	G
G	—	empty

DFS (from A, recursion/stack)

Visit	Why
A	start
B	first neighbor of A
D	first unvisited neighbor of B
E	D is done → back to B, then E
F	unvisited neighbor of E
C	first unvisited neighbor of F
G	C is done → back to F, then G

BFS order: A B C D E F G (levels: A=0; B, C=1; D, E, F=2; G=3 — gives the shortest path in an unweighted graph) · **DFS order:** A B D E F C G · both are O(V + E).

Adjacency list vs matrix — which is faster at what Combined 5 Bank AP 2023

Matrix: "Is there an edge between u and v?" — O(1); good for dense graphs; space $O(V^2)$.

List: "All neighbors of u" or BFS/DFS — O(deg u) and O(V + E) in total; good for sparse graphs; space O(V + E).

4.9 Dijkstra — shortest path

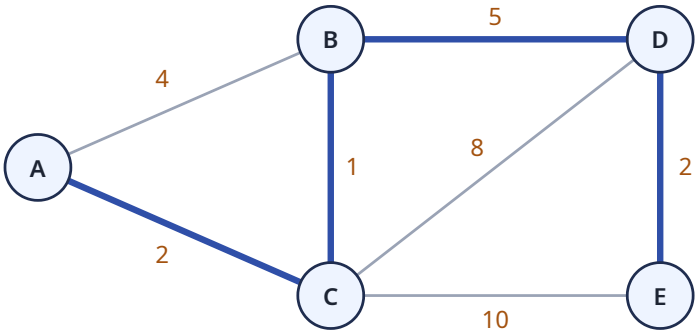


Figure 4.3: Weighted undirected graph (thick blue = MST, see 4.10)

Step	Chosen (finalized)	A	B	C	D	E
0	—	0	∞	∞	∞	∞
1	A (0)	0	4 (A)	2 (A)	∞	∞
2	C (2)	0	**3 (C)**	2	10 (C)	12 (C)
3	B (3)	0	3	2	**8 (B)**	12
4	D (8)	0	3	2	8	**10 (D)**
5	E (10)	0	3	2	8	10

Result: A→C = 2; A→B = 3 (A–C–B); A→D = 8 (A–C–B–D); A→E = 10 (A–C–B–D–E). The predecessor is in brackets. With a binary heap it is O((V + E) log V); Dijkstra does not work with negative weights (use Bellman-Ford).

4.10 Minimum Spanning Tree — Kruskal and Prim (Figure 4.3)

Kruskal (smallest edge first, skip if it forms a cycle)		
Edge	**Weight**	**Decision**
B–C	1	Take
A–C	2	Take
D–E	2	Take
A–B	4	Skip (A–C–B cycle)
B–D	5	Take → 4 edges, stop

Prim (from A, cheapest outgoing edge)		
In the tree	**Edge taken**	**Weight**
{A}	A–C	2
{A, C}	C–B	1
{A, B, C}	B–D	5
{A, B, C, D}	D–E	2

Both methods give MST = {B–C, A–C, D–E, B–D}, total weight **10**; V − 1 = 4 edges. Kruskal is O(E log E) (sort + union-find). Prim is O(E log V) (heap).

4.11 Topological sort (Kahn's algorithm)

DAG: A→C, B→C, B→D, C→E, D→E, E→F. In-degree: A 0, B 0, C 2, D 1, E 2, F 1.

Step	Removed from queue	In-degree update	Queue
Start	—	in-degree 0: A, B	A B
1	A	C: 2→1	B
2	B	C: 1→0, D: 1→0	C D
3	C	E: 2→1	D
4	D	E: 1→0	E
5	E	F: 1→0	F
6	F	—	Empty

Order: **A B C D E F** (B A D C E F is also valid — a topological order can be non-unique). If not all nodes are removed, the graph has a cycle. O(V + E). Uses: course prerequisites, build dependencies, task scheduling.

4.12 Sorting — every pass, same array [29, 10, 14, 37, 13]

Algorithm	Pass 1	Pass 2	Pass 3	Pass 4
Bubble	[10, 14, 29, 13, **37**]	[10, 14, 13, **29, 37**]	[10, 13, **14, 29, 37**]	No swap → stop
Selection	min 10 ↔ 29: [**10**, 29, 14, 37, 13]	min 13 ↔ 29: [10, **13**, 14, 37, 29]	min 14 (already in place)	min 29 ↔ 37: [10, 13, 14, 29, 37]
Insertion	key 10: [**10, 29**, 14, 37, 13]	key 14: [10, 14, 29, 37, 13]	key 37: no change	key 13: [10, 13, 14, 29, 37]

Total swaps in Bubble = 5 = number of inversions in the array ((29,10), (29,14), (29,13), (14,13), (37,13)).

Merge sort

```
[29, 10, 14, 37, 13]
split → [29, 10, 14] | [37, 13]
split → [29, 10] [14] | [37] [13]
split → [29] [10]
merge → [10, 29]
merge → [10, 14, 29]       [13, 37]
merge → [10, 13, 14, 29, 37]
```

Quick sort (Lomuto, pivot = last element)

```
[29, 10, 14, 37, 13], pivot 13, i = -1
 j=0: 29 > 13
 j=1: 10 ≤ 13 → i=0, swap a[0],a[1] → [10,
29, 14, 37, 13]
 j=2: 14 > 13;  j=3: 37 > 13
 swap a[i+1], pivot → [10, 13, 14, 37, 29]
(pivot index 1)
Right part [14, 37, 29], pivot 29:
 14 ≤ 29 → stays in place; 37 > 29
 swap → [14, 29, 37]
Result: [10, 13, 14, 29, 37]
```

When is Quick sort's worst case $O(n^2)$?

When the array is already sorted (or reverse sorted) and the pivot is always the last/first element. Then every partition splits into (n − 1) and 0, the depth is n, and total comparisons are n(n − 1)/2. Fix: random pivot or median-of-three.

4.13 Binary search trace

Sorted array (index 0–9): [3, 8, 15, 21, 27, 34, 42, 56, 63, 77]; mid = ⌊(lo + hi)/2⌋.

key = 63 (present)			
lo	**hi**	**mid**	**a[mid]**
0	9	4	27 < 63 → lo = 5
5	9	7	56 < 63 → lo = 8
8	9	8	63 = key → index 8

key = 20 (absent)			
lo	**hi**	**mid**	**a[mid]**
0	9	4	27 > 20 → hi = 3
0	3	1	8 < 20 → lo = 2
2	3	2	15 < 20 → lo = 3
3	3	3	21 > 20 → hi = 2; lo > hi → −1

Maximum comparisons $\lceil \log_2(n + 1) \rceil$: n = 10 → 4; n = 10^6 → 20.

4.14 Infix → Postfix and Prefix, and Postfix evaluation

Expression: **(A + B) * C − D / E**. Precedence: ^ > * / > + −; for equal precedence, left to right (except ^).

Token	Stack	Output	Remark
(	(		push
A	(	A	operand → output
+	(+	A	
B	(+	A B	
)		A B +	pop until (
*	*	A B +	
C	*	A B + C	
−	−	A B + C *	pop * (higher/equal precedence), then push −
D	−	A B + C * D	
/	− /	A B + C * D	/ has higher precedence → push
E	− /	A B + C * D E	
End		A B + C * D E / −	empty the stack

Postfix: A B + C * D E / − · **Prefix:** − * + A B C / D E (reverse the expression, swap the brackets, get the postfix, then reverse again; or directly: the main operator − comes first).

Evaluation: A = 2, B = 3, C = 4, D = 10, E = 5

Postfix 2 3 + 4 * 10 5 / −: stack → [2] → [2, 3] → + → [5] → [5, 4] → * → [20] → [20, 10] → [20, 10, 5] → / → [20, 2] → − → **[18]**. Check: (2 + 3) × 4 − 10/5 = 20 − 2 = 18 ✔

Extra practice: `6 2 3 + - 3 8 2 / + * 2 ^ 3 +` → 6−5 = 1; 8/2 = 4; 3+4 = 7; 1×7 = 7; 7^2 = 49; 49+3 = **52**.

Rule: when an operator arrives, pop two — the first popped is the right operand, the second is the left (order matters in a − b).

4.15 Queue using two stacks BSCS AP 17.01.2026

Logic

Enqueue(x): push onto S1. **Dequeue():** if S2 is empty, pop all elements from S1 one by one and push them onto S2 (the order reverses; the oldest element ends up on top of S2); then pop from S2. If S2 is not empty, pop directly from S2. Each element moves at most twice → amortized O(1).

Operation	S1 (bottom → top)	S2 (bottom → top)	Output
enqueue 1	1	—	
enqueue 2	1 2	—	
enqueue 3	1 2 3	—	
dequeue	—	3 2 (1 popped)	1
enqueue 4	4	3 2	
dequeue	4	3	2
dequeue	4	—	3
dequeue	—	— (4 moved, then popped)	4

Output 1, 2, 3, 4 — elements came out in the order they went in (FIFO) ✔

```
void enqueue(int x){ push(&S1, x); }
int dequeue(void){
    if (isEmpty(&S2))
        while (!isEmpty(&S1)) push(&S2, pop(&S1));   /* reverse it */
    if (isEmpty(&S2)) { printf("Queue empty\n"); return -1; }
    return pop(&S2);
}
```

4.16 Page replacement — FIFO, LRU, Optimal Combined 5 Bank AP 2023

Reference string: **1 2 3 4 2 3 4 1 2 1 1 3 1 4**, 3 frames. F = page fault, — = hit.

Ref	1	2	3	4	2	3	4	1	2	1	1	3	1	4
LRU — replace the least recently used page														
Frame 1	1	1	1	4	4	4	4	4	4	4	4	3	3	3

Frame 2		2	2	2	2	2	2	1	1	1	1	1	1	1
Frame 3			3	3	3	3	3	3	2	2	2	2	2	4
Fault	F	F	F	F	—	—	—	F	F	—	—	F	—	F
FIFO — replace the page that entered first														
Frame 1	1	1	1	4	4	4	4	4	4	4	4	3	3	3
Frame 2		2	2	2	2	2	2	1	1	1	1	1	1	4
Frame 3			3	3	3	3	3	3	2	2	2	2	2	2
Fault	F	F	F	F	—	—	—	F	F	—	—	F	—	F
Optimal — replace the page needed farthest in the future (or never again)														
Frame 1	1	1	1	4	4	4	4	1	1	1	1	1	1	4
Frame 2		2	2	2	2	2	2	2	2	2	2	2	2	2
Frame 3			3	3	3	3	3	3	3	3	3	3	3	3
Fault	F	F	F	F	—	—	—	F	—	—	—	—	—	F

Page faults: LRU = 8, FIFO = 8, Optimal = 6 (hit ratio 6/14, 6/14, 8/14 respectively).

LRU's hard decisions, explained

Ref 4 (4th): among 1, 2, 3, page 1 was used longest ago → replace 1.
Ref 1 (8th): last uses — 2 (5th), 3 (6th), 4 (7th) → replace 2.
Ref 2 (9th): 3 (6th), 4 (7th), 1 (8th) → replace 3.
Ref 3 (12th): 4 (7th), 2 (9th), 1 (11th) → replace 4.
Ref 4 (14th): 2 (9th), 3 (12th), 1 (13th) → replace 2.

Belady's anomaly

In FIFO, adding frames can sometimes *increase* faults. Reference 1 2 3 4 1 2 5 1 2 3 4 5: 9 faults with 3 frames, 10 faults with 4 frames. LRU and Optimal are "stack algorithms", so they do not have this problem. Optimal is not practical (it needs knowledge of the future) — it is only a benchmark for comparison.

4.17 Complexity cheat-sheet

Data structure / Algorithm	Access / Search	Insert	Delete	Remark
Array	O(1) / O(n)	O(n)	O(n)	Insert at end O(1) (dynamic array, amortized)
Singly linked list	O(n)	O(1) at start	O(1) at start	Insert at end is O(1) if a tail pointer exists
Stack / Queue	—	O(1)	O(1)	LIFO / FIFO

Hash table	Average O(1), worst O(n)	Average O(1)	Average O(1)	Worst case when all keys go to the same slot
BST	Average O(log n), worst O(n)	Same	Same	O(n) if skewed
AVL / Red-Black	O(log n)	O(log n)	O(log n)	Always balanced
Binary heap	max/min O(1)	O(log n)	extract O(log n)	build O(n)
Binary search	O(log n)	—	—	Needs a sorted array
BFS / DFS	O(V + E) (list), O(V^2) (matrix)			BFS → unweighted shortest path
Dijkstra	O((V + E) log V) with a heap			No negative weights
Kruskal / Prim	O(E log E) / O(E log V)			MST
Merge / Heap sort	O(n log n) in all cases			Merge is stable, heap is in-place
Quick sort	Average O(n log n), worst O(n^2)			In-place, fastest in practice
Bubble / Insertion / Selection	O(n^2) (bubble with flag and insertion: best case O(n))			Small data
Tower of Hanoi / naive Fibonacci	O(2^n)			Exponential

Chapter 5 — 10-Mark Theory: Ready Answers

In the BIBM AP paper, about half of the 150 marks come from direct theory questions — "Explain…", "Differentiate…", "Compare…". The same question types keep repeating in AP 2024 (10198), AP 03.01.2026 and AP 17.01.2026 (Inheritance, Polymorphism, SDLC, Waterfall vs Agile, TCP vs UDP, OSI, Process vs Thread, Hashing vs Encryption, CIA, Cloud models). This chapter gives an **exam-ready English answer, to be written as it is,** for each one — definition, table, example. Each answer can be written in 10–12 minutes.

> **Answer-Writing Formula**
> Definition (1–2 lines) → table/diagram → example (a banking one if possible) → one-line conclusion. For a "Differentiate" question, give a table of at least 5 rows.

5.1 Object-Oriented Programming

Q1. What are the four pillars of OOP? AP 17.01.2026 AP 2024 (10198)

Pillar	Meaning	How achieved (Java)	Example
Encapsulation	Binding data and methods in one unit and hiding internal state	`private` fields + public getters/setters	`Account.balance` changed only via `deposit()`
Abstraction	Showing only essential features, hiding implementation	abstract class, interface	ATM shows "Withdraw" — not the CBS logic
Inheritance	A class acquires properties of another (IS-A)	`extends`, `implements`	`SavingsAccount extends Account`
Polymorphism	One interface, many forms	Overloading (compile-time), overriding (run-time)	`calculateInterest()` differs per account type

Tip: Remember the four names as "A PIE" — Abstraction, Polymorphism, Inheritance, Encapsulation.

Q2. Abstraction vs Encapsulation

Basis	Abstraction	Encapsulation
Focus	*What* an object does	*How* data is protected
Level	Design level	Implementation level
Hides	Complexity / implementation details	Data (internal state)
Achieved by	Abstract class, interface	Access modifiers (private/protected), getters/setters
Example	A `Payment` interface with `pay()`	`private double balance;` with validated `withdraw()`

Q3. Abstract class vs Interface (Java)

Basis	Abstract class	Interface
Methods	Abstract and concrete	Abstract by default; `default` / `static` methods since Java 8
Variables	Any type (instance, static, final or not)	Only `public static final` constants
Constructor	Yes	No
Inheritance	A class extends only one	A class can implement many (multiple inheritance of type)
Access modifiers	Any	Methods implicitly public
Use when	Related classes share code/state (IS-A)	Unrelated classes share a capability (CAN-DO), e.g. `Comparable`

Q4. What is a constructor? Explain its types.

A **constructor** is a special member function with the **same name as the class** and **no return type**, invoked automatically when an object is created, used to initialize the object.

- **Default constructor** — no parameters; supplied by the compiler if none is written.
- **Parameterized constructor** — takes arguments to initialize fields with given values.
- **Copy constructor** — creates an object from another object of the same class (built-in in C++; written manually in Java).
- **Constructor overloading** — several constructors with different parameter lists; `this(...)` chains them.

```
class Account {
    String no; double balance;
    Account() { this("N/A", 0); }                 // default
    Account(String no, double b) { this.no = no; this.balance = b; } // parameterized
    Account(Account a) { this(a.no, a.balance); }  // copy
}
```

Q5. Static (early) vs Dynamic (late) binding; Overloading vs Overriding

Basis	Static binding / Overloading	Dynamic binding / Overriding
Resolved at	Compile time	Run time (by actual object type)
Polymorphism type	Compile-time	Run-time
Where	Same class (or subclass)	Parent–child classes
Signature	Same name, **different parameters**	Same name, **same parameters** and compatible return type

Applies to	`static`, `private`, `final` methods are statically bound	Only instance (non-static, non-final, non-private) methods
Example	`add(int,int)`, `add(double,double)`	`Shape s = new Circle(); s.area();` calls `Circle.area()`

> **Trap**
> A static method is not overridden — writing a static method with the same signature in a subclass is **method hiding**. Replacing an instance method with a static one (or the reverse) gives a compile error (BB AP 2023, A/B question).

Q6. Explain inheritance and its types. AP 17.01.2026 BSCS 2020

Inheritance lets a child (sub) class acquire fields and methods of a parent (super) class, enabling **code reuse**, extensibility and an IS-A hierarchy.

Type	Structure	Java support
Single	B → A	Yes
Multilevel	C → B → A	Yes
Hierarchical	B → A, C → A	Yes
Multiple	C → A and C → B	Only through interfaces (avoids diamond problem)
Hybrid	Combination of the above	Through interfaces

Example: `Account` → `SavingsAccount`, `CurrentAccount` (hierarchical). Advantages: reuse, less duplication, method overriding for specialised behaviour, easier maintenance.

Q7. What is polymorphism? Describe its types. AP 2024 (10198)

Polymorphism ("many forms") is the ability of the same method call or operator to behave differently depending on the object or arguments.

1. **Compile-time (static) polymorphism** — method overloading, operator overloading (C++). Decided by the compiler from the argument list.
2. **Run-time (dynamic) polymorphism** — method overriding with a superclass reference to a subclass object; resolved by dynamic method dispatch.

```
class Account { double rate() { return 0.0; } }
class Savings extends Account { double rate() { return 0.04; } }
class Fixed   extends Account { double rate() { return 0.08; } }
Account a = new Fixed();
System.out.println(a.rate());   // 0.08 -- decided at run time
```

Q8. SOLID principles (brief)

Letter	Principle	One-line meaning
S	Single Responsibility	A class should have only one reason to change
O	Open/Closed	Open for extension, closed for modification

L	Liskov Substitution	Subclass objects must be usable wherever the parent is expected
I	Interface Segregation	Many small specific interfaces are better than one fat interface
D	Dependency Inversion	Depend on abstractions, not on concrete classes

5.2 Database Management System

Q1. Types of keys in DBMS

Key	Definition	Example (Customer)
Super key	Any set of attributes that uniquely identifies a tuple	{NID}, {NID, Name}
Candidate key	Minimal super key	{NID}, {AccountNo}
Primary key	Chosen candidate key; unique and NOT NULL	AccountNo
Alternate key	Candidate keys not chosen as primary	NID
Foreign key	Attribute referring to the primary key of another table; ensures referential integrity	Transaction.AccountNo
Composite key	Key made of 2+ attributes	(StudentID, CourseID)
Surrogate key	System-generated identifier with no business meaning	AUTO_INCREMENT id

Q2. Explain normalization (1NF to BCNF) with an example.

Normalization is the process of organising tables to reduce redundancy and avoid insertion, update and deletion anomalies, by decomposing them based on functional dependencies.

Start: `Enroll(StudentID, StudentName, CourseID, CourseName, Teacher, TeacherPhone, Grade)` where one row holds "C101, C102" in a cell.

Normal form	Rule	Action on example
1NF	Atomic values; no repeating groups	One row per (StudentID, CourseID); key = (StudentID, CourseID)
2NF	1NF + no **partial dependency** (non-key depends on whole key)	StudentName depends only on StudentID, CourseName only on CourseID ⇒ split: Student(StudentID, StudentName), Course(CourseID, CourseName, Teacher, TeacherPhone), Enroll(StudentID, CourseID, Grade)
3NF	2NF + no **transitive dependency** (non-key → non-key)	CourseID → Teacher → TeacherPhone ⇒ Teacher(Teacher, TeacherPhone); Course(CourseID, CourseName, Teacher)
BCNF	For every FD X → Y, X is a super key	If a teacher teaches only one course and (Student, Course) → Teacher but Teacher → Course, split into (Teacher, Course) and (Student, Teacher)

Tip: 2NF = "depends on the whole key", 3NF = "depends on nothing but the key", BCNF = "the left side is always a super key".

Q3. ACID properties with a banking example

Property	Meaning	Fund transfer Tk 5,000 from A to B
Atomicity	All or nothing	Debit A and credit B both happen, or neither (ROLLBACK on failure)
Consistency	DB moves from one valid state to another; constraints hold	Total money unchanged; `CHECK (balance >= 0)` never violated
Isolation	Concurrent transactions do not see each other's intermediate state	Another query never sees A debited but B not yet credited
Durability	Committed changes survive crashes	After COMMIT, the transfer persists even after power failure (redo log)

Q4. DDL, DML, DCL and TCL BSCS 2020

Category	Purpose	Commands
DDL — Data Definition Language	Define/modify structure (auto-commit)	CREATE, ALTER, DROP, TRUNCATE, RENAME
DML — Data Manipulation Language	Work with data rows	SELECT*, INSERT, UPDATE, DELETE, MERGE
DCL — Data Control Language	Permissions	GRANT, REVOKE
TCL — Transaction Control Language	End or manage a transaction	**COMMIT** (make permanent), **ROLLBACK** (undo), **SAVEPOINT** (partial rollback point), SET TRANSACTION

*Many people treat SELECT separately as DQL (Data Query Language).

Q5. DELETE vs TRUNCATE vs DROP

Basis	DELETE	TRUNCATE	DROP
Type	DML	DDL	DDL
Removes	Selected rows (WHERE allowed)	All rows	Whole table (structure + data + indexes)
Rollback	Yes (before commit)	Generally no (auto-commit in most DBMS)	No
Triggers	Fires row triggers	Does not fire	Does not fire
Speed	Slower (row-by-row logging)	Faster (deallocates pages)	Fastest
Identity counter	Not reset	Reset	Gone with table

Q6. WHERE vs HAVING

WHERE	HAVING
Filters rows **before** grouping	Filters groups **after** GROUP BY
Cannot use aggregate functions	Used with aggregates (SUM, AVG, COUNT)
Works with SELECT, UPDATE, DELETE	Only with SELECT

```
SELECT DeptCode, AVG(Salary) AS avg_sal
FROM Employee
WHERE Salary > 10000          -- row filter
GROUP BY DeptCode
HAVING AVG(Salary) > 50000;    -- group filter
```

Q7. Index — advantages and disadvantages

An **index** is a separate data structure (usually a B+ tree or hash) on one or more columns that lets the DBMS locate rows without scanning the whole table.

Advantages

- Faster SELECT, WHERE, JOIN, ORDER BY
- Enforces uniqueness (UNIQUE index)
- Reduces disk I/O

Disadvantages

- Extra storage
- Slower INSERT/UPDATE/DELETE (index must be updated)
- Useless on low-cardinality columns (e.g. gender)

Clustered index = physical order of rows (one per table); non-clustered = separate structure with pointers (many allowed).

Q8. Types of relationships in DBMS; Book–Borrower case AP 2024 (10198) AP 17.01.2026

Relationship	Meaning	Example	Implementation
One-to-One (1:1)	One A ↔ one B	Person – NID	FK with UNIQUE in either table
One-to-Many (1:N)	One A ↔ many B	Branch – Accounts	FK on the "many" side
Many-to-Many (M:N)	Many A ↔ many B	Student – Course	Junction table with composite key

Library case: One borrower borrows many books, and over time the same book is borrowed by many borrowers ⇒ **Many-to-Many**. Resolve with `Loan(LoanID, BookID FK, BorrowerID FK, IssueDate, DueDate, ReturnDate)`. If we consider a single physical copy *at one moment*, it is held by at most one borrower, so Borrower–(current) Copy is 1:N — state this assumption in the answer.

Tip: If asked for the types of "Relation", mention both the relationship cardinality (1:1, 1:N, M:N) and, if needed, the unary/binary/ternary degree — in one line.

Q9. Mapping an ER diagram to tables

1. **Strong entity** → table; attributes → columns; key → primary key.
2. **Weak entity** → table with owner's PK as FK; PK = owner PK + partial key.
3. **1:N relationship** → put PK of the "1" side as FK in the "N" side table.

4. **1:1** → FK (with UNIQUE) in either side, preferably the total-participation side.
5. **M:N** → new table containing both PKs (composite PK) plus relationship attributes.
6. **Multivalued attribute** → separate table (entity PK, value).
7. **Composite attribute** → store only its simple components.

Q10. Deadlock in a database

A **deadlock** occurs when two or more transactions each hold a lock the other needs and wait forever. Example: T1 locks Account A and waits for B; T2 locks B and waits for A.

- **Detection:** build a *wait-for graph*; a cycle ⇒ deadlock; abort a victim (youngest / least work) and roll it back.
- **Prevention:** timestamp schemes — *wait-die* (older waits, younger dies) and *wound-wait* (older wounds/aborts younger); or lock all resources in a fixed order.
- **Timeout:** abort a transaction that waits longer than a threshold.

5.3 Operating System

Q1. Process vs Thread; advantages of threads AP 17.01.2026

Basis	Process	Thread
Definition	A program in execution with its own address space	Smallest unit of CPU scheduling inside a process
Memory	Separate address space	Shares code, data, heap, open files of its process; has own stack, registers, PC
Creation/switch cost	Heavy	Light
Communication	IPC needed (pipes, sockets, shared memory)	Directly through shared memory
Failure isolation	One crash does not affect others	One faulty thread can crash the whole process

Advantages of threads: (1) faster creation and context switch, (2) resource sharing without IPC, (3) responsiveness — UI stays responsive while a background thread works, (4) better use of multi-core CPUs.

Q2. Process states

New → Ready → Running → (Waiting/Blocked) → Ready … → Terminated.

- New → Ready: admitted by long-term scheduler.
- Ready → Running: dispatched by short-term scheduler.
- Running → Ready: time quantum expired / preempted.
- Running → Waiting: I/O or event wait.
- Waiting → Ready: I/O completed.
- Running → Terminated: exit.

Each process is represented by a **PCB** (PID, state, PC, registers, memory info, open files, priority).

Q3. Types of scheduling and scheduling algorithms BSCS 2020

Scheduler	Role
Long-term (job)	Selects jobs from disk into memory; controls degree of multiprogramming
Short-term (CPU)	Selects a ready process for the CPU; runs very frequently
Medium-term	Swaps processes out/in to reduce memory load

Algorithm	Preemptive?	Key point
FCFS	No	Simple; convoy effect
SJF	No	Minimum average waiting time; needs burst estimate; starvation
SRTF	Yes	Preemptive SJF
Priority	Both	Starvation solved by **aging**
Round Robin	Yes	Time quantum; fair; good for time-sharing
Multilevel (feedback) queue	Yes	Separate queues per class; processes may move between queues

Q4. Deadlock — necessary conditions and handling

Deadlock can occur only if all four hold simultaneously (Coffman conditions):

1. **Mutual exclusion** — a resource is non-shareable.
2. **Hold and wait** — a process holds one resource and waits for another.
3. **No preemption** — resources cannot be forcibly taken.
4. **Circular wait** — P1 waits for P2, P2 for P3 … Pn for P1.

Strategy	How
Prevention	Break one condition, e.g. request all resources at once, or number resources and request in increasing order (no circular wait)
Avoidance	Grant only if the system stays in a **safe state** — Banker's algorithm
Detection & recovery	Resource-allocation/wait-for graph; kill a process or preempt resources
Ignore (Ostrich)	Used by most general OSs when deadlock is rare

Q5. Semaphore vs Mutex

Basis	Mutex	Semaphore
Type	Locking object (binary)	Integer signalling variable (counting or binary)
Ownership	Only the thread that locked can unlock	Any process can signal
Operations	lock() / unlock()	wait() (P, decrement) / signal() (V, increment)
Allows	One thread in critical section	Up to N threads (counting semaphore)
Use	Protect a shared variable	Manage a pool of identical resources, ordering

Q6. Producer–Consumer problem AP 17.01.2026

A **producer** puts items into a **bounded buffer** of size N and a **consumer** removes them. Issues to solve: (1) **mutual exclusion** on the buffer (race condition on the count/index), (2) producer must **wait when the buffer is full**, (3) consumer must **wait when it is empty**. Solution with three semaphores:

```
semaphore mutex = 1, empty = N, full = 0;

Producer:                    Consumer:
while (true) {               while (true) {
  item = produce();            wait(full);
  wait(empty);                 wait(mutex);
  wait(mutex);                 item = remove();
  insert(item);                signal(mutex);
  signal(mutex);               signal(empty);
  signal(full);                consume(item);
}                            }
```

Tip: Placing wait(mutex) before wait(empty) can cause deadlock — mentioning this order earns extra marks.

Q7. Paging vs Segmentation

Basis	Paging	Segmentation
Unit	Fixed-size pages/frames	Variable-size logical segments (code, stack, data)
View	Physical/system view	Programmer's logical view
Fragmentation	Internal	External
Address	(page no., offset)	(segment no., offset)
Table	Page table (frame no.)	Segment table (base, limit)

Q8. Virtual memory and thrashing

Virtual memory lets a process execute even if it is not fully in RAM; pages are loaded on demand (**demand paging**) and the disk acts as backing store. Benefits: programs larger than RAM, more processes in memory, isolation. A **page fault** occurs when a referenced page is not in memory; a replacement algorithm (FIFO, LRU, Optimal) chooses a victim.

Thrashing — the system spends more time swapping pages than executing, because processes do not have enough frames for their working set; CPU utilisation collapses. Remedies: working-set model, page-fault-frequency control, reducing the degree of multiprogramming, adding RAM.

Q9. Boot process of a computer — correct order AP 03.01.2026

1. **Power on → BIOS/UEFI POST** (Power-On Self Test checks CPU, RAM, devices).
2. **MBR/GPT lookup** — firmware finds the boot device and loads the bootloader (GRUB / Windows Boot Manager).
3. **OS kernel load** — bootloader loads the kernel into memory.
4. **Driver initialization** — kernel initialises drivers and starts system services (init/systemd).
5. **User login** — login screen / shell.

5.4 Networking

Q1. OSI layers and functions; OSI vs TCP/IP AP 03.01.2026

#	OSI layer	Function	PDU	Protocols/devices	TCP/IP layer
7	Application	Network services to user apps	Data	HTTP, FTP, SMTP, DNS	Application
6	Presentation	Translation, encryption, compression		SSL/TLS, JPEG, ASCII	
5	Session	Establish/maintain/terminate sessions, sync		NetBIOS, RPC	
4	Transport	End-to-end delivery, segmentation, flow/error control, ports	Segment / Datagram	TCP, UDP	Transport
3	Network	Logical addressing, routing	Packet	IP, ICMP, OSPF; router	Internet
2	Data Link	Framing, MAC addressing, error detection	Frame	Ethernet, PPP; switch	Network Access
1	Physical	Bits over medium; signals	Bit	Cable, hub, repeater	

OSI is a 7-layer reference model; TCP/IP is a 4-layer practical model used on the Internet (OSI's top three merge into Application; bottom two into Network Access/Link).

Memory sentence (7→1): "All People Seem To Need Data Processing".

Q2. Compare TCP and UDP with examples AP 03.01.2026

Basis	TCP	UDP
Connection	Connection-oriented (3-way handshake)	Connectionless
Reliability	Reliable: ACK, retransmission	Unreliable, best effort
Ordering	In-order delivery (sequence numbers)	No ordering
Flow/congestion control	Yes (sliding window)	No
Header	20–60 bytes	8 bytes
Speed	Slower	Faster, low latency
Examples	HTTP/HTTPS, FTP, SMTP, SSH, internet banking	DNS query, VoIP, live video streaming, online games, DHCP

Q3. How does TCP make an unreliable network reliable? BSCS 2020

- **Sequence numbers** — every byte is numbered; receiver reorders and **discards duplicates**.

- **Acknowledgements** — cumulative ACK confirms receipt.
- **Retransmission** — on timeout or 3 duplicate ACKs (fast retransmit), lost segments are resent.
- **Checksum** — detects corrupted segments, which are dropped and retransmitted.
- **Flow control** — receiver window prevents overflow of the receiver buffer.
- **Congestion control** — slow start, congestion avoidance, fast recovery.
- **Connection management** — 3-way handshake to open, 4-way FIN exchange to close.

Q4. TCP 3-way handshake

1. Client → Server: **SYN** (seq = x)
2. Server → Client: **SYN-ACK** (seq = y, ack = x + 1)
3. Client → Server: **ACK** (ack = y + 1) — connection ESTABLISHED.

Termination: FIN → ACK → FIN → ACK (4 segments). SYN flood attack abuses step 2 by never sending step 3 (mitigated by SYN cookies).

Q5. IPv4 vs IPv6

Basis	IPv4	IPv6
Address size	32 bit (≈ 4.3 billion)	128 bit (≈ 3.4×10^{38})
Notation	Dotted decimal 192.168.1.1	Hexadecimal colon 2001:db8::1
Header	Variable 20–60 B, has checksum	Fixed 40 B, no checksum
Address resolution	ARP	NDP (Neighbor Discovery, ICMPv6)
Configuration	Manual / DHCP	SLAAC or DHCPv6
Broadcast	Yes	No (multicast/anycast)
Security	IPsec optional	IPsec built into the design

Q6. Subnetting — quick method

Subnets = $2^{\text{borrowed bits}}$ · Hosts/subnet = $2^{\text{host bits}} - 2$ · Block size = 256 − (interesting octet of mask)

Example: Split 192.168.10.0/24 into 4 subnets

Borrow 2 bits ⇒ /26, mask 255.255.255.192, block = 64, hosts = 62.

Subnet	Network	Usable range	Broadcast
1	192.168.10.0	.1 – .62	.63
2	192.168.10.64	.65 – .126	.127
3	192.168.10.128	.129 – .190	.191
4	192.168.10.192	.193 – .254	.255

Q7. DNS, DHCP and ARP

Protocol	Purpose	Port / layer	How it works
DNS	Domain name → IP	UDP/TCP 53	Resolver → root → TLD (.com) → authoritative server; answers cached by TTL
DHCP	Automatic IP, mask, gateway, DNS	UDP 67 (server) / 68 (client)	**DORA**: Discover → Offer → Request → Acknowledge
ARP	IP → MAC on the LAN	Layer 2/3	Broadcast "Who has 192.168.1.1?"; owner unicasts its MAC; result cached in ARP table

Q8. OSPF vs RIP; how packets are delivered with OSPF AP 17.01.2026

Basis	RIP	OSPF
Type	Distance vector	Link state
Algorithm	Bellman-Ford	Dijkstra (SPF)
Metric	Hop count (max 15)	Cost (based on bandwidth)
Updates	Full table every 30 s	LSAs only on change (triggered)
Convergence	Slow (count-to-infinity)	Fast
Scale	Small networks	Large, hierarchical (areas, backbone area 0)

Packet delivery with OSPF: (1) routers form *neighbour adjacencies* with Hello packets; (2) each router floods **LSAs** describing its links and costs; (3) every router builds an identical **link-state database** (map of the network); (4) each runs **Dijkstra** with itself as root to compute the shortest-path tree; (5) best next hops go into the **routing table**; (6) for each packet the router reads the destination IP, does a **longest-prefix match**, decrements TTL and forwards it hop by hop until it reaches the destination network, where ARP finds the host MAC.

Q9. Hub vs Switch vs Router

Basis	Hub	Switch	Router
OSI layer	1 (Physical)	2 (Data Link)	3 (Network)
Address used	None	MAC	IP
Forwarding	Broadcasts to all ports	To specific port via MAC table	Between networks via routing table
Collision domain	One shared	One per port	One per interface
Broadcast domain	One	One (unless VLANs)	Separates broadcast domains
Use	Obsolete	LAN inside a branch	Branch ↔ head office / Internet

5.5 Information Security

Q1. CIA triad AP 03.01.2026 BB AP 2023

Principle	Meaning	Banking example	Controls
Confidentiality	Only authorised people can read data	Customer balance hidden from others	Encryption, access control, MFA
Integrity	Data is accurate and unaltered	Transfer amount cannot be modified in transit	Hashing, digital signature, checksums, audit logs
Availability	Systems accessible when needed	ATM/internet banking 24×7	Redundancy, backup, DR site, DDoS protection

Q2. Authentication vs Authorization AP 03.01.2026

Basis	Authentication	Authorization
Question	"Who are you?"	"What are you allowed to do?"
Order	First	After authentication
Based on	Password, OTP, fingerprint, certificate	Roles, permissions, ACLs, policies
Example	Teller logs in with ID + password + OTP	Teller may post deposits but cannot approve loans
Protocols	Kerberos, OpenID Connect	OAuth 2.0 scopes, RBAC

Q3. Hashing vs Encryption AP 03.01.2026

Basis	Hashing	Encryption
Direction	One-way	Two-way (decrypt with key)
Key	No key (HMAC uses one)	Needs key(s)
Output length	Fixed (SHA-256 = 256 bits)	Roughly proportional to input
Goal	Integrity, password storage	Confidentiality
Examples	SHA-256, SHA-3, bcrypt	AES, RSA

Q4. Symmetric vs Asymmetric encryption

Basis	Symmetric	Asymmetric
Keys	One shared secret key	Public + private key pair
Speed	Fast — bulk data	Slow — small data, key exchange
Key distribution	Hard (must share securely)	Easy (public key is public)
Examples	AES, DES, 3DES	RSA, ECC, Diffie-Hellman
Use	Disk/database encryption	Digital signatures, TLS handshake

TLS (HTTPS) combines both: asymmetric to agree a session key, symmetric (AES) to encrypt the data.

Q5. Digital signature — how it works

1. Sender computes hash H = SHA-256(message).
2. Sender encrypts H with **sender's private key** → signature.
3. Message + signature are sent.
4. Receiver decrypts the signature with **sender's public key** → H_1, and computes H_2 = hash(received message).
5. $H_1 = H_2 \Rightarrow$ **integrity**, **authentication** and **non-repudiation** are ensured.

Q6. SQL Injection, XSS and CSRF — prevention

Attack	How it works	Prevention
SQL Injection	Attacker input becomes part of SQL, e.g. `' OR '1'='1`	Prepared statements/parameterised queries, input validation, least-privilege DB account, WAF
XSS	Malicious script injected into a page runs in victims' browsers (steals cookies)	Output encoding/escaping, Content Security Policy, HttpOnly cookies, input sanitisation
CSRF	Victim's logged-in browser is tricked into sending an unwanted request	Anti-CSRF tokens, SameSite cookies, re-authentication for critical actions

Q7. Social engineering AP 03.01.2026

Social engineering is manipulating people (not systems) into revealing confidential information or performing unsafe actions.

- **Phishing** (email), **vishing** (voice call: "I am from the bank, tell me your OTP"), **smishing** (SMS)
- **Pretexting** (fake identity/story), **baiting** (infected USB), **tailgating** (following into a secure area), **quid pro quo**

Prevention: security awareness training, verify requests by a second channel, never share OTP/PIN, MFA, email filtering, clear reporting procedure.

Q8. Vulnerability Assessment vs Penetration Testing AP 03.01.2026

Basis	Vulnerability Assessment	Penetration Testing
Goal	Find and list as many weaknesses as possible	Exploit weaknesses to prove real impact
Approach	Mostly automated scanning (Nessus, OpenVAS)	Manual + tools (Metasploit, Burp Suite)
Depth	Breadth-first	Depth-first
Output	Prioritised vulnerability list (CVSS)	Attack narrative, proof of exploitation
Frequency	Frequent (monthly/quarterly)	Periodic (yearly / after major change)

Q9. Firewall vs IDS vs IPS

Basis	Firewall	IDS	IPS
Function	Allows/blocks traffic by rules (IP, port, protocol, app)	Detects suspicious activity and alerts	Detects and blocks automatically

Placement	Network perimeter, between zones	Out-of-band (SPAN/mirror port)	Inline
Action	Preventive	Passive / detective	Active / preventive
Method	Packet filter, stateful, NGFW	Signature + anomaly	Signature + anomaly

Q10. New employee account creation with security best practices AP 03.01.2026

1. **Authorised request** — HR/line manager submits an approved access request form; ID verified.
2. **Unique account** — one user ID per person (no shared IDs); naming convention.
3. **Role-based access (RBAC)** & **least privilege** — assign only the roles needed (e.g. Teller); maker-checker for sensitive functions.
4. **Password policy** — minimum 8–12 characters with upper/lower/digit/special; no reuse of last N passwords; change every 60–90 days; lockout after 3–5 failed attempts; temporary password must be changed at first login.
5. **MFA** for remote, privileged and CBS access.
6. **Secure credential delivery** — separate channel; never by plain email.
7. **Awareness** — user signs the ICT security policy/acceptable-use policy.
8. **Logging & review** — creation logged; periodic access review; on transfer/exit, **offboarding**: disable account immediately, revoke tokens/VPN, collect devices.

Q11. Explicit Deny vs Explicit Allow AP 03.01.2026

User X ∈ Group A and Group B. Folder 1: Read → Group A; **explicit Deny Write → Group B**; Write → User X (explicit allow).

Answer: No, User X cannot write. In access-control systems such as NTFS, permissions from all groups are combined and an **explicit Deny overrides an explicit Allow** at the same level. Since X is a member of Group B, the Deny Write applies to X even though X was granted Write individually. X can still **Read** (from Group A). To allow writing, remove X from Group B or remove the Deny entry.

5.6 Software Engineering

Q1. SDLC and its phases AP 17.01.2026

The **Software Development Life Cycle** is a structured process to plan, build, test, deliver and maintain software with predictable quality, cost and time.

Phase	Main activity	Output
1. Planning & feasibility	Scope, cost, schedule, risk	Project plan
2. Requirement analysis	Gather functional/non-functional needs	SRS document
3. Design	Architecture, database, UI, modules	Design document (HLD/LLD)
4. Implementation	Coding, code review	Source code
5. Testing	Unit, integration, system, acceptance	Test reports
6. Deployment	Release to production	Live system

7. Maintenance	Bug fixes, enhancements	Updated versions

Q2. Limitations of Waterfall and how Agile addresses them AP 03.01.2026

Waterfall limitation	Agile solution
Requirements frozen early; change is costly	Welcomes change; backlog re-prioritised every sprint
Working software only at the end	Working increment every 1–4 week sprint
Testing late ⇒ defects found late	Continuous testing, CI, TDD
Little customer involvement after requirements	Product owner/customer reviews each sprint
High risk for long/uncertain projects	Short feedback loops reduce risk
Heavy documentation	"Working software over comprehensive documentation"

Waterfall still suits small projects with stable, well-understood requirements (e.g. regulatory reports with fixed format).

Q3. Scrum — roles, events and artifacts

Roles	Events	Artifacts
Product Owner (maximises value, owns backlog) Scrum Master (coach, removes impediments) Developers (build the increment)	Sprint (1–4 weeks) Sprint Planning Daily Scrum (15 min) Sprint Review Sprint Retrospective	Product Backlog Sprint Backlog Increment (meets Definition of Done)

Q4. Functional vs non-functional requirements; requirement validation AP 03.01.2026

Basis	Functional	Non-functional
Describes	What the system does	How well it does it (quality attributes)
Example	"User can transfer funds to another account"	"Transfer completes within 2 s; 99.9% availability; AES-256 encryption"
Testing	Functional testing	Performance, security, usability testing
If missing	Feature absent	System slow/insecure/unusable

Requirement validation checks that the documented requirements actually define the system the customer wants. Checks: validity, consistency, completeness, realism, verifiability. Techniques: **requirement reviews**, **prototyping**, **test-case generation**, traceability analysis, walkthroughs with stakeholders.

Q5. Testing levels and types BSCS 2020

Level	Tests	By
Unit testing	Individual function/class in isolation (JUnit)	Developer

Integration testing	Interfaces between modules (top-down, bottom-up, big-bang)	Developer/tester
System testing	Whole system against SRS (functional + non-functional)	QA team
Acceptance testing (UAT)	Business readiness — alpha/beta	Customer/users

Other types: **regression** (old features still work after change), **smoke** (basic build check), **performance/load/stress**, **security**, **usability**.

Q6. Black-box vs White-box testing

Basis	Black-box	White-box
Knowledge of code	Not required	Required
Based on	Requirements/specification	Internal logic, paths
Techniques	Equivalence partitioning, boundary value analysis, decision table	Statement, branch, path coverage; cyclomatic complexity
Level	System, acceptance	Unit, integration
Done by	Testers/users	Developers

Q7. Types of UML diagrams

Structural (static)	Behavioural (dynamic)
Class, Object, Component, Deployment, Package, Composite structure	Use case, Activity, State machine, Sequence, Communication, Timing, Interaction overview

Class-diagram relations: association (—), aggregation (◇ hollow diamond, "has-a", part can exist alone), composition (◆ filled diamond, part dies with whole), generalization (△ hollow triangle, IS-A), dependency (- - →), multiplicity (1, 0..1, *, 1..*).

Q8. Git — commit vs push and other commands AP 17.01.2026

Version control records changes to files over time so that teams can collaborate, track who changed what, and roll back. Git is a distributed VCS — every developer has the full history locally.

Command	What it does	Where
`git commit`	Saves a snapshot of staged changes with a message	**Local** repository only
`git push`	Uploads local commits to a remote (GitHub/GitLab)	Local → **remote**
`git pull`	Fetch + merge remote changes	Remote → local
`git branch` / `checkout -b`	Create/switch an independent line of development	Local
`git merge`	Combine another branch into the current one	Local
`git clone`	Copy a remote repository	Remote → local

Memory tip: commit = writing it in your own notebook; push = sharing it with everyone. You can commit offline; push needs a remote.

5.7 Web, Cloud and AI

Q1. Cloud computing — service and deployment models AP 03.01.2026

Cloud computing is on-demand delivery of computing resources (servers, storage, databases, software) over the Internet with pay-as-you-go pricing. NIST characteristics: on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service.

Service model	Provider manages	User manages	Examples
IaaS	Hardware, virtualization, network	OS, runtime, apps, data	AWS EC2, Azure VM
PaaS	+ OS, runtime, middleware	Apps and data	Google App Engine, Heroku
SaaS	Everything	Only usage/config	Gmail, Microsoft 365, Salesforce

Deployment model	Description
Public	Shared infrastructure owned by provider (AWS, Azure)
Private	Dedicated to one organisation (a bank's own data center cloud)
Hybrid	Private + public combined (core banking private, website public)
Community	Shared by organisations with common concerns (e.g. several banks)

Q2. Virtualization, hypervisor, VM vs container AP 03.01.2026

Virtualization creates multiple virtual instances (machines, storage, networks) on one physical resource. The **hypervisor** (VMM) sits between hardware and VMs, allocating CPU, memory and I/O and isolating VMs.

- **Type 1 (bare-metal):** runs directly on hardware — VMware ESXi, Microsoft Hyper-V, KVM, Xen.
- **Type 2 (hosted):** runs on a host OS — VirtualBox, VMware Workstation.

Basis	Virtual Machine	Container
Virtualizes	Hardware	Operating system
Contains	Full guest OS + app	App + libraries; shares host kernel
Size	GBs	MBs
Boot time	Minutes	Seconds
Isolation	Strong	Process-level (weaker)
Tools	VMware, Hyper-V	Docker, Kubernetes (orchestration)

Q3. REST API principles

- **Client–server** separation; **stateless** requests (each carries all needed info, e.g. token).
- **Uniform interface** — resources identified by URIs (`/accounts/123`), manipulated with HTTP methods.

- **Cacheable** responses, **layered system**, optional code-on-demand.
- Representations usually **JSON**; status codes convey results.

Q4. HTTP methods and status codes

Method	Use	Idempotent
GET	Read a resource (data in URL)	Yes
POST	Create / submit data (data in body)	No
PUT	Replace a resource	Yes
PATCH	Partial update	Not necessarily
DELETE	Remove a resource	Yes

Class	Meaning	Common codes
1xx	Informational	100 Continue
2xx	Success	200 OK, 201 Created, 204 No Content
3xx	Redirection	301 Moved Permanently, 302 Found, 304 Not Modified
4xx	Client error	400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found
5xx	Server error	500 Internal Server Error, 502 Bad Gateway, 503 Service Unavailable

Q5. Supervised vs Unsupervised vs Reinforcement learning AP 17.01.2026

Basis	Supervised	Unsupervised	Reinforcement
Data	Labelled (input → known output)	Unlabelled	No dataset; agent interacts with environment
Goal	Predict label/value	Find hidden structure/patterns	Learn a policy maximising cumulative reward
Tasks	Classification, regression	Clustering, dimensionality reduction, association	Sequential decision making
Algorithms	Linear/logistic regression, decision tree, SVM, neural networks	K-means, hierarchical clustering, PCA, Apriori	Q-learning, SARSA, policy gradient
Feedback	Direct (correct answer)	None	Delayed reward/penalty
Banking example	Loan default prediction, fraud classification	Customer segmentation	Dynamic pricing, trading agents

Q6. Why can a heuristic search (A*) be faster than a blind search (BFS)? AP 17.01.2026

- **BFS** is uninformed: it expands nodes level by level in all directions, without knowing where the goal is. Time and memory grow as $O(b^d)$ (b = branching factor, d = depth of goal).
- **A*** evaluates **f(n) = g(n) + h(n)**: g = cost so far, h = estimated cost to goal. It always expands the node that looks closest to the goal, so it explores far fewer nodes — a good heuristic "points" the search

towards the goal.

- If h is **admissible** (never overestimates) A* is still optimal; the better h approximates the true cost, the fewer nodes are expanded.
- In practice, faster heuristics may be **inadmissible** or weighted (f = g + w·h, w > 1). These trade a guaranteed optimal path for speed — the path is "good enough" but not always perfect.
- **Argument:** search cost depends on the number of nodes expanded; domain knowledge (h) prunes large parts of the search space that BFS must visit; therefore A* reaches the goal sooner even if, with an approximate heuristic, the path is occasionally sub-optimal.

Example: in route finding on a map, straight-line distance as h makes A* head towards the destination city, while BFS expands every road in every direction.

Q7. Overfitting

Overfitting: the model learns noise and details of the training data, giving high training accuracy but poor accuracy on new data (high variance). **Underfitting**: model too simple; poor on both (high bias).

Remedies: more training data, cross-validation, regularization (L1/L2), dropout, early stopping, pruning (trees), feature selection, simpler model, data augmentation.

5.8 Digital Logic, Data Communication and Language Processor

Q1. Number system conversion — quick method BSCS 2020

$(10010.101)_2 = (?)_{10}$

Integer: 1×16 + 0 + 0 + 1×2 + 0 = 18 · Fraction: 1×½ + 0×¼ + 1×⅛ = 0.5 + 0.125 = 0.625 ⇒ **18.625**

$(543)_{10} = (?)_{16}$

543 ÷ 16 = 33 rem 15 (F); 33 ÷ 16 = 2 rem 1; 2 ÷ 16 = 0 rem 2 ⇒ read upward: **21F**. Check: 2×256 + 1×16 + 15 = 543 ✔

Conversion	Method
Decimal → base b (integer)	Repeated division by b; remainders read bottom-up
Decimal → base b (fraction)	Repeated multiplication by b; integer parts read top-down
Binary ↔ Octal	Group 3 bits from the binary point
Binary ↔ Hex	Group 4 bits from the binary point

Q2. Universal gate — prove NAND is universal AP 17.01.2026

A **universal gate** can implement any Boolean function without using any other gate type. NAND and NOR are universal. Since {AND, OR, NOT} is functionally complete, it is enough to build these three from NAND:

Gate	NAND construction	Proof
NOT	Tie both inputs: NAND(A, A)	(A·A)' = A'

AND	NAND followed by NAND-NOT: NAND(NAND(A,B), NAND(A,B))	((AB)')' = AB
OR	NAND(NAND(A,A), NAND(B,B))	(A'·B')' = A + B (De Morgan)

Gate count: NOT = 1, AND = 2, OR = 3 NAND gates. Hence any circuit can be built using only NAND gates.

Q3. Alarm truth table: Alarm = (Motion AND Night) OR Panic AP 03.01.2026

M (motion)	N (night)	P (panic)	M·N	Alarm = M·N + P
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Alarm = Σm(1, 3, 5, 6, 7). Circuit: one 2-input AND (M, N) feeding a 2-input OR with P.

Q4. Checksum, Parity and CRC AP 17.01.2026

Checksum logic: the sender divides data into k-bit words (e.g. 16 bits), adds them using **one's-complement arithmetic** (wrap-around carry), complements the sum and sends it as the checksum. The receiver adds all words *including* the checksum; if the result is all 1s (complement = 0) the data is accepted, otherwise it is corrupted.

Example (4-bit words): data 1001, 0110, 1100

1001 + 0110 = 1111; 1111 + 1100 = 1 1011 → wrap carry: 1011 + 1 = 1100 · checksum = complement = **0011**.

Receiver: 1001 + 0110 + 1100 + 0011 = 1100 + 0011 = 1111 → complement 0000 ⇒ no error ✔

File transfer: the source publishes a cryptographic checksum (e.g. SHA-256 hash) of the file; after download the user recomputes it (`sha256sum file.iso`). Equal values ⇒ the file is complete and unaltered; any single changed bit gives a totally different hash.

Method	Detects	Used in
Parity bit (even/odd)	All single-bit (odd number of) errors; misses even-count errors	Memory, serial links
Checksum	Most errors; weak for reordering	IP, TCP, UDP headers
CRC	All burst errors ≤ r bits (r = degree of generator); very strong	Ethernet, Wi-Fi, storage
Hamming code	Detects 2-bit, **corrects** 1-bit errors	ECC memory

Q5. Compiler vs Interpreter vs Assembler AP 03.01.2026

Basis	Compiler	Interpreter	Assembler
Input	High-level source (whole program)	High-level source (line/statement at a time)	Assembly language
Output	Object/executable file	No separate file; executes directly	Machine code
Speed of execution	Fast	Slower	Fast
Error reporting	After analysing the whole program	Stops at the first error	After assembly
Memory	More (object code)	Less	—
Examples	C, C++ (gcc), Go	Python, JavaScript, Ruby	NASM, MASM

Java uses both: `javac` compiles to bytecode, the JVM interprets/JIT-compiles it.

Chapter 6 — AP MCQ Bank (150)

The MCQ section of Job ID-25106 carries 100 marks (Subject 75 + Math 15 + GK 10) in 1 hour — an average of 36 seconds per question. The questions below follow the pattern of the BIBM Combined Bank Officer (IT) MCQ 2026 and past AP papers.

> **Negative Marking (−0.25) Strategy**
>
> 0.25 is deducted for each wrong answer. If you can **definitely eliminate at least one option**, guessing pays off (expected marks +0.17); eliminating two gives +0.375. If you cannot eliminate any, skip the question. Answer the sure questions in the first round and the rest in the second round.

> **How to Practice**
>
> Set a 30-minute timer for every 50 questions. Cover the answers, then check at the end and calculate (correct × 1 − wrong × 0.25).

6.1 Programming and OOP (1–20)

1. What is the output of `printf("%d", 7 / 2);` in C?

(a) 3.5 (b) 3.0
(c) 4 (d) 3

Answer: (d) — Division of two ints is integer division; the decimal part is discarded.

2. What is the output of `System.out.println(5 + 3 + "AB");` in Java?

(a) Compile error (b) 8AB
(c) 53AB (d) AB8

Answer: (b) — Evaluated left to right: 5 + 3 = 8, then "AB" is appended.

3. What is the output of `System.out.println("AB" + 5 + 3);` in Java?

(a) AB 5 3 (b) AB53
(c) 8AB (d) AB8

Answer: (b) — The first operand is a String, so the following + operators are concatenation.

4. What is the output of `int x = 5; printf("%d", x << 2);` in C?

(a) 20 (b) 7
(c) 25 (d) 10

Answer: (a) — Left shift by 2 = $\times 2^2 = 5 \times 4 = 20$.

5. What is the output of the code below? `int i; for(i = 0; i < 5; i++); printf("%d", i);`

(a) 0 1 2 3 4 (b) 5
(c) Nothing (d) 4

Answer: (b) — There is a semicolon after the for, so the loop body is empty; at the end i = 5 and printf runs once.

6. In C, if `int a[5] = {1, 2};`, what is the value of a[3]?

(a) Garbage
(b) 2
(c) Error
(d) 0

Answer: (d) — In partial initialization, the remaining elements become 0.

7. If `int f(int n){ return n == 0 ? 0 : n + f(n - 1); }`, then f(4) = ?

(a) 4
(b) 24
(c) 10
(d) 6

Answer: (c) — 4 + 3 + 2 + 1 + 0 = 10.

8. What is the output of `print(2 ** 3 ** 2)` in Python?

(a) 64
(b) 36
(c) 12
(d) 512

Answer: (d) — ** is evaluated right to left: $3^2 = 9$, $2^9 = 512$.

9. What is the output of `printf("%d", sizeof(char));` in C?

(a) 8
(b) 2
(c) 4
(d) 1

Answer: (d) — By the C standard, sizeof(char) is always 1.

10. Which of the following is NOT one of the four pillars of OOP?

(a) Compilation
(b) Encapsulation
(c) Polymorphism
(d) Inheritance

Answer: (a) — The four pillars: Abstraction, Encapsulation, Inheritance, Polymorphism.

11. What is it called when data and methods are bound into one unit and the data is hidden?

(a) Abstraction
(b) Encapsulation
(c) Modularity
(d) Polymorphism

Answer: (b) — Appeared almost word for word in the BIBM Officer IT MCQ 2026.

12. When is method overloading resolved?

(a) Link time
(b) Load time
(c) Compile time
(d) Run time

Answer: (c) — It is compile-time (static) polymorphism; overriding is resolved at run time.

13. Which keyword prevents a class from being inherited in Java?

(a) static
(b) abstract
(c) private
(d) final

Answer: (d) — A `final class` cannot be extended.

14. How is multiple inheritance of classes achieved in Java?

(a) Not possible by any means
(b) Using an abstract class
(c) Using a package
(d) Using an interface

Answer: (d) — A class can implement multiple interfaces.

15. What is the return type of a constructor?

(a) No return type
(b) The class type
(c) int
(d) void

Answer: (a) — No return type is written for a constructor, not even void.

16. Which of the following is NOT a primitive data type in Java?

(a) char
(b) boolean
(c) int
(d) String

Answer: (d) — String is a class (reference type).

17. What is the default value of the Java instance variable `int count;`?

(a) −1
(b) 0
(c) Garbage
(d) null

Answer: (b) — The default of an instance/static numeric field is 0 (a local variable has no default).

18. Which access modifier makes a member visible only inside the same class?

(a) public
(b) private
(c) default
(d) protected

Answer: (b)

19. What is it called when a subclass defines a static method with the same signature as the superclass?

(a) Method hiding
(b) Overloading
(c) Overriding
(d) Shadowing error

Answer: (a) — A static method is not overridden; it is hidden.

20. In C, `int x = 10; int y = (x > 5) ? x * 2 : x / 2; printf("%d", y);`

(a) 10
(b) 20
(c) 5
(d) 15

Answer: (b) — The condition is true, so x * 2 = 20.

6.2 Data Structures and Algorithms (21–40)

21. Which principle does a stack follow?

(a) Priority
(b) Random
(c) FIFO
(d) LIFO

Answer: (d) — Last In First Out.

22. Which data structure is needed to check balanced parentheses?

(a) Graph
(b) Tree
(c) Stack
(d) Queue

Answer: (c) — Push an opening bracket; on a closing bracket, pop and match.

23. Which data structure is generally used to implement recursion?

(a) Hash table
(b) Stack
(c) Queue
(d) Heap

Answer: (b) — The function call stack.

24. What is the time complexity of binary search on a sorted array?

(a) O(n)
(b) O(n log n)
(c) O(log n)
(d) O(1)

Answer: (c) — The search space is halved at each step.

25. For 1,000,000 (10^6) sorted elements, about how many comparisons does binary search need at most?

(a) 20
(b) 10
(c) 1000
(d) 500000

Answer: (a) — $2^{20} \approx 10^6$, so ≈ 20 comparisons.

26. What is the worst-case time complexity of quick sort?

(a) O(n)
(b) O(n log n)
(c) O(log n)
(d) $O(n^2)$

Answer: (d) — When the pivot is always the smallest/largest (e.g., first element as pivot on sorted input).

27. Which sorting algorithm is O(n log n) in all cases and stable?

(a) Selection sort
(b) Heap sort
(c) Merge sort
(d) Quick sort

Answer: (c) — Heap sort is O(n log n) but not stable.

28. What is the best-case complexity of optimized bubble sort with a swap flag?

(a) $O(n^2)$
(b) O(1)
(c) O(n)
(d) O(n log n)

Answer: (c) — If already sorted, there is no swap in one pass and it stops.

29. Which traversal of a Binary Search Tree gives elements in sorted order?

(a) Preorder
(b) Inorder
(c) Level order
(d) Postorder

Answer: (b) — Left → Root → Right.

30. What is the complexity of adding a new node at the beginning of a singly linked list?

(a) $O(n^2)$ (b) O(log n)
(c) O(n) (d) O(1)

Answer: (d) — Only the head pointer needs to change.

31. Which data structure is used in BFS (Breadth First Search)?

(a) Queue (b) Heap
(c) Stack (d) Tree

Answer: (a) — DFS uses a stack (or recursion).

32. What is the postfix form of the infix `A + B * C`?

(a) +A*BC (b) AB+C*
(c) AB*C+ (d) ABC*+

Answer: (d) — * has higher precedence: B C * first, then A is added.

33. What is the value of the postfix `2 3 4 * +`?

(a) 20 (b) 14
(c) 24 (d) 9

Answer: (b) — 3 × 4 = 12, then 2 + 12 = 14.

34. In which case may Dijkstra's algorithm fail to give the correct result?

(a) Negative weight edge (b) Directed graph
(c) Weighted graph (d) Disconnected graph

Answer: (a) — Bellman-Ford is needed for negative edges.

35. Which of the following is NOT a collision resolution method in hashing?

(a) Quadratic probing (b) Binary search
(c) Linear probing (d) Chaining

Answer: (b)

36. Hash table size 13, h(k) = k mod 13, linear probing; 3 is already present. In which slot will 16 go?

(a) 4 (b) 16
(c) 3 (d) 5

Answer: (a) — 16 mod 13 = 3, occupied ⇒ next empty slot is 4 (part of a BB AP 2023 question).

37. How many edges does a spanning tree of a connected graph with n vertices have?

(a) n − 1 (b) n + 1
(c) n (d) n(n − 1)/2

Answer: (a)

38. What is the complexity of extracting the root (maximum) from a binary max-heap?

(a) O(n log n) (b) O(1)
(c) O(n) (d) O(log n)

Answer: (d) — Move the last element to the root and heapify down; the height is log n.

39. Which dynamic programming approach starts from small subproblems and fills a table?

(a) Top-down (b) Backtracking
(c) Bottom-up (d) Greedy

Answer: (c) — Bottom-up, or tabulation (BIBM 2026 MCQ).

40. If a queue is built using two stacks, what is the amortized complexity of enqueue?

(a) O(1) (b) O(log n)
(c) $O(n^2)$ (d) O(n)

Answer: (a) — Each element is moved at most twice, so O(1) on average (idea from the AP written question of 17.01.2026).

6.3 DBMS and SQL (41–58)

41. Which clause filters groups using an aggregate after GROUP BY?

(a) HAVING (b) WHERE
(c) WITH (d) ORDER BY

Answer: (a)

42. Which command deletes both the structure and the data of a table?

(a) DELETE (b) REMOVE
(c) DROP (d) TRUNCATE

Answer: (c)

43. TRUNCATE belongs to which category of commands?

(a) DML (b) DCL
(c) TCL (d) DDL

Answer: (d)

44. GRANT and REVOKE belong to which category?

(a) DML (b) TCL
(c) DCL (d) DDL

Answer: (c) — Data Control Language.

45. Which command ends a transaction and makes the changes permanent?

(a) COMMIT (b) END
(c) ROLLBACK (d) SAVE

Answer: (a) — ROLLBACK cancels the changes; both are TCL.

46. What does "A" in ACID stand for?

(a) Accuracy
(b) Availability
(c) Atomicity
(d) Authorization

Answer: (c) — All or nothing.

47. Which statement is true for a primary key?

(a) A table has multiple primary keys
(b) It can be duplicate
(c) It can be NULL
(d) Unique and NOT NULL

Answer: (d)

48. What does a foreign key ensure?

(a) Domain integrity
(b) Entity integrity
(c) Referential integrity
(d) Normalization

Answer: (c)

49. Which dependency does 2NF remove?

(a) Multivalued
(b) Join
(c) Transitive
(d) Partial

Answer: (d) — 3NF removes transitive dependency.

50. How is a many-to-many relationship implemented?

(a) Using an index
(b) Using a view
(c) Using an FK in one table
(d) Using a junction (associative) table

Answer: (d) — A composite key made of the PKs of the two tables.

51. Which join returns all rows of the left table and the matching rows of the right table?

(a) RIGHT JOIN
(b) CROSS JOIN
(c) INNER JOIN
(d) LEFT JOIN

Answer: (d) — If there is no match, the right-side columns are NULL.

52. Which keyword removes duplicate rows from the result?

(a) SINGLE
(b) DISTINCT
(c) DIFFERENT
(d) UNIQUE

Answer: (b)

53. Which wildcard in the LIKE operator represents any number of characters?

(a) *
(b) _
(c) %
(d) ?

Answer: (c) — _ means exactly one character.

54. Which is correct for sorting students by marks from highest to lowest?

(a) ORDER BY marks DESC
(b) ORDER BY marks ASC
(c) SORT BY marks DESC
(d) GROUP BY marks DESC

Answer: (a)

55. What does `COUNT(*)` count?

(a) Only non-NULL values
(b) All rows, including NULL
(c) Only distinct values
(d) Number of columns

Answer: (b) — COUNT(column) skips NULL.

56. Which of the following is NOT an aggregate function?

(a) COUNT
(b) AVG
(c) SUM
(d) UPPER

Answer: (d) — UPPER is a scalar (string) function.

57. What generally happens when an index is created?

(a) Storage decreases
(b) INSERT is faster, SELECT is slower
(c) SELECT is faster, INSERT/UPDATE is slightly slower
(d) All operations are slower

Answer: (c)

58. The `CHECK (balance >= 0)` constraint helps protect which ACID property?

(a) Isolation
(b) Durability
(c) Atomicity
(d) Consistency

Answer: (d) — The DB does not accept a transaction that breaks the rule.

6.4 Operating System (59–72)

59. Which scheduling algorithm gives the minimum average waiting time?

(a) SJF
(b) FCFS
(c) Priority
(d) Round Robin

Answer: (a)

60. What is the key parameter of Round Robin scheduling?

(a) Burst estimate
(b) Priority
(c) Time quantum
(d) Arrival order only

Answer: (c)

61. Which of the following do threads of the same process NOT share?

(a) Stack
(b) Heap
(c) Code
(d) Open files

Answer: (a) — Each thread has its own stack and registers.

62. How many conditions must hold together for a deadlock to occur?

(a) 3
(b) 5
(c) 4
(d) 2

Answer: (c) — Mutual exclusion, hold & wait, no preemption, circular wait.

63. What is Banker's algorithm used for?

(a) Deadlock avoidance
(b) Deadlock detection
(c) CPU scheduling
(d) Deadlock prevention

Answer: (a) — It grants resources after checking for a safe state.

64. What does thrashing mean?

(a) More time spent on paging than on execution
(b) Deadlock
(c) Disk becoming full
(d) CPU overheating

Answer: (a)

65. In which page replacement algorithm is Belady's anomaly seen?

(a) LFU
(b) LRU
(c) Optimal
(d) FIFO

Answer: (d) — Page faults can increase even when frames are increased.

66. The producer–consumer problem is mainly which type of problem?

(a) Disk scheduling
(b) Paging
(c) Process synchronization
(d) Memory allocation

Answer: (c) — Bounded buffer; solved with a mutex + empty + full semaphore.

67. What is the condition called that prevents two processes from entering the critical section at the same time?

(a) Starvation
(b) Race condition
(c) Synchronous condition
(d) Mutual exclusion

Answer: (d) — BIBM 2026 MCQ.

68. Where does context switching occur?

(a) In the compiler
(b) On the hard disk
(c) In the CPU
(d) In the I/O device

Answer: (c) — The CPU moves from one process to another, and the state is saved in the PCB.

69. Which is the correct order of the computer boot process?

(a) MBR → POST → Login → Kernel
(b) Drivers → POST → Kernel → Login
(c) Kernel → POST → Bootloader → Login
(d) POST → MBR/GPT → Kernel → Drivers → Login

Answer: (d) — Written question of AP 03.01.2026.

70. Which command changes file permissions in Linux?

(a) chown
(b) chgrp
(c) chmod
(d) umask

Answer: (c) — chown changes the owner.

71. After `chmod 644 file`, what is the group's permission?

(a) rw-
(b) r--
(c) r-x
(d) rwx

Answer: (b) — 6 = rw- (owner), 4 = r-- (group), 4 = r-- (others).

72. In which state is a process that is waiting for the CPU but is in memory?

(a) New
(b) Terminated
(c) Ready
(d) Waiting

Answer: (c) — If it is waiting for I/O, it is Waiting/Blocked.

6.5 Networking (73–90)

73. How many layers does the OSI model have?

(a) 5
(b) 8
(c) 7
(d) 4

Answer: (c)

74. Routing is the job of which OSI layer?

(a) Data Link
(b) Session
(c) Transport
(d) Network

Answer: (d)

75. What is the PDU name of TCP at the Transport layer?

(a) Packet
(b) Bit
(c) Segment
(d) Frame

Answer: (c) — Network: packet, Data link: frame.

76. What is the default port of HTTPS?

(a) 8080
(b) 443
(c) 22
(d) 80

Answer: (b)

77. What is the standard TCP port of SSH?

(a) 21
(b) 25
(c) 23
(d) 22

Answer: (d) — 21 FTP, 23 Telnet, 25 SMTP.

78. Which port does DNS use?

(a) 110
(b) 143
(c) 53
(d) 67

Answer: (c)

79. Which protocol is used to receive email?

(a) SNMP
(b) POP3
(c) FTP
(d) SMTP

Answer: (b) — IMAP also receives; SMTP sends.

80. How many bits are IPv4 and IPv6 addresses, respectively?

(a) 64 and 128
(b) 32 and 64
(c) 16 and 32
(d) 32 and 128

Answer: (d)

81. Which is a private IP address?

(a) 172.32.1.1
(b) 192.168.10.5
(c) 11.0.0.1
(d) 8.8.8.8

Answer: (b) — Private: 10.0.0.0/8, 172.16.0.0–172.31.255.255, 192.168.0.0/16.

82. How many usable hosts are in a /26 subnet?

(a) 126
(b) 62
(c) 30
(d) 64

Answer: (b) — 6 host bits: $2^6 - 2 = 62$.

83. What is the subnet mask of /27?

(a) 255.255.255.192
(b) 255.255.255.240
(c) 255.255.255.224
(d) 255.255.255.128

Answer: (c) — Three 1s in the last octet: 128 + 64 + 32 = 224.

84. What is the correct order of the four DHCP steps?

(a) Discover–Request–Offer–Ack
(b) Offer–Discover–Ack–Request
(c) Discover–Offer–Request–Ack
(d) Request–Offer–Discover–Ack

Answer: (c) — DORA.

85. What does ARP do?

(a) IP → MAC
(b) Assigns IP
(c) MAC → Port
(d) Domain → IP

Answer: (a)

86. What is the correct order of the TCP 3-way handshake?

(a) SYN, SYN-ACK, ACK
(b) SYN-ACK, SYN, ACK
(c) ACK, SYN, FIN
(d) SYN, ACK, SYN-ACK

Answer: (a)

87. Which connectionless protocol is widely used in real-time streaming?

(a) UDP
(b) TCP
(c) FTP
(d) SMTP

Answer: (a) — Low overhead, no retransmission.

88. Which algorithm does OSPF use?

(a) Kruskal
(b) Prim
(c) Dijkstra (SPF)
(d) Bellman-Ford

Answer: (c) — RIP uses Bellman-Ford, max hop 15.

89. Which device (Layer 2) forwards frames by looking at the MAC address?

(a) Router
(b) Switch
(c) Hub
(d) Repeater

Answer: (b)

90. Ping to 8.8.8.8 works but www.google.com does not open — what is the likely cause?

(a) DHCP failure
(b) Cable disconnected
(c) DNS problem
(d) No gateway

Answer: (c) — IP connectivity exists; name resolution is not working (BIBM 2026 MCQ).

6.6 Security (91–102)

91. In the CIA triad, what does "I" stand for?

(a) Integrity
(b) Isolation
(c) Information
(d) Identity

Answer: (a)

92. Which principle prevents a sender from later denying that they sent a message?

(a) Availability
(b) Confidentiality
(c) Non-repudiation
(d) Integrity

Answer: (c) — Ensured by a digital signature.

93. Which is an asymmetric encryption algorithm?

(a) RSA
(b) DES
(c) AES
(d) 3DES

Answer: (a)

94. Which is a cryptographic hash function?

(a) RSA-2048
(b) AES-256
(c) SHA-256
(d) Base64

Answer: (c) — Base64 is only an encoding.

95. "This is the bank calling, tell me your OTP" — which attack is this?

(a) Social engineering (vishing)
(b) DDoS
(c) SQL injection
(d) Brute force

Answer: (a)

96. What is the most effective way to prevent SQL injection?

(a) Antivirus
(b) Using HTTPS
(c) Prepared statement / parameterized query
(d) Long password

Answer: (c)

97. Which part of the CIA triad does a DoS/DDoS attack mainly hit?

(a) Confidentiality
(b) Availability
(c) Integrity
(d) Authentication

Answer: (b)

98. Which one detects suspicious traffic and automatically blocks it inline?

(a) Hub
(b) Proxy
(c) IPS
(d) IDS

Answer: (c) — IDS only raises an alert.

99. What is the main purpose of a DMZ?

(a) Storing backups
(b) Assigning IPs
(c) Increasing LAN speed
(d) Keeping public-facing servers in a separate, secure zone

Answer: (d)

100. What is the use of Password + OTP + fingerprint called?

(a) Encryption
(b) Multi-factor authentication
(c) Authorization
(d) Single sign-on

Answer: (b) — know + have + are.

101. Which principle gives a user only the minimum access needed for the job?

(a) Least privilege
(b) Separation of duties
(c) Zero day
(d) Defense in depth

Answer: (a)

102. What type of control is security awareness training?

(a) Detective only
(b) Technical
(c) Administrative
(d) Physical

Answer: (c) — BIBM 2026 MCQ.

6.7 SE, Web, Cloud and AI (103–110)

103. Which is the linear sequential SDLC model?

(a) Waterfall
(b) Agile
(c) Spiral
(d) Prototype

Answer: (a)

104. Which Git command sends local commits to the remote repository?

(a) git commit
(b) git pull
(c) git push
(d) git clone

Answer: (c)

105. What does HTTP status 404 mean?

(a) Internal Server Error
(b) Forbidden
(c) Bad Request
(d) Not Found

Answer: (d)

106. Which cloud service model are Heroku and Google App Engine?

(a) IaaS
(b) DaaS
(c) PaaS
(d) SaaS

Answer: (c)

107. Which hypervisor runs directly on the hardware (bare-metal)?

(a) Type 2
(b) Container
(c) Type 1
(d) Emulator

Answer: (c) — For example VMware ESXi, Hyper-V.

108. Which type of learning groups customers into clusters from data without labels (clustering)?

(a) Reinforcement
(b) Supervised
(c) Transfer
(d) Unsupervised

Answer: (d)

109. What is the evaluation function of A* search?

(a) f(n) = g(n)
(b) f(n) = h(n)
(c) f(n) = g(n) × h(n)
(d) f(n) = g(n) + h(n)

Answer: (d) — With h(n) alone it is greedy best-first search.

110. What is testing done only against the requirements, without looking at the code, called?

(a) Code review
(b) Black-box testing
(c) Unit testing
(d) White-box testing

Answer: (b)

6.8 Mathematics (111–135)

111. 20% of 250 + 15% of 400 = ?

(a) 105
(b) 110
(c) 120
(d) 100

Answer: (b) — 50 + 60 = 110.

112. A number is increased by 20% and then decreased by 20%. What is the net change?

(a) No change
(b) 4% decrease
(c) 4% increase
(d) 2% decrease

Answer: (b) — 1.2 × 0.8 = 0.96.

113. Cost price is 800 taka and selling price is 920 taka. What is the profit percentage?

(a) 20%
(b) 18%
(c) 15%
(d) 12%

Answer: (c) — 120/800 × 100 = 15%.

114. A product costs 1000 taka. What is the selling price after successive discounts of 10% and 5%?

(a) 850
(b) 860
(c) 845
(d) 855

Answer: (d) — 1000 × 0.9 × 0.95 = 855.

115. If a 20% discount still gives a 20% profit, by what % is the marked price above the cost price?

(a) 20%
(b) 40%
(c) 30%
(d) 50%

Answer: (d) — CP 100, SP 120 = 0.8 MP ⇒ MP = 150 (BIBM 2026 MCQ).

116. If A : B = 2 : 3 and B : C = 4 : 5, then A : B : C = ?

(a) 2 : 3 : 5
(b) 8 : 12 : 15
(c) 4 : 6 : 5
(d) 8 : 12 : 10

Answer: (b) — Make B = 12: A = 8, C = 15.

117. A does a job in 12 days and B in 18 days. In how many days will they finish it together?

(a) 7.5 days
(b) 15 days
(c) 7.2 days
(d) 6 days

Answer: (c) — 1/12 + 1/18 = 5/36 ⇒ 36/5 = 7.2.

118. 36 km/h = how many m/s?

(a) 6
(b) 18
(c) 12
(d) 10

Answer: (d) — 36 × 5/18 = 10.

119. A boat's speed in still water is 10 km/h and the current is 2 km/h. How long to go 60 km downstream?

(a) 4 hours
(b) 7.5 hours
(c) 5 hours
(d) 6 hours

Answer: (c) — Downstream speed is 12 km/h; 60/12 = 5.

120. A 150 m long train passes a pole in 10 seconds. What is the train's speed?

(a) 60 km/h
(b) 15 km/h
(c) 54 km/h
(d) 45 km/h

Answer: (c) — 15 m/s × 18/5 = 54 km/h.

121. If $(a + b)^2 = 49$ and $ab = 6$, then $a^2 + b^2 = ?$

(a) 61
(b) 37
(c) 43
(d) 55

Answer: (b) — $49 - 2 \times 6 = 37$.

122. If $x - 1/x = 3$, then $x^2 + 1/x^2 = ?$

(a) 7
(b) 9
(c) 13
(d) 11

Answer: (d) — $(x - 1/x)^2 + 2 = 9 + 2 = 11$.

123. If $a + b = 5$ and $a - b = 1$, then $a^2 - b^2 = ?$

(a) 4
(b) 24
(c) 6
(d) 5

Answer: (d) — $(a + b)(a - b) = 5 \times 1$.

124. What is the area of a circle with radius 7 cm ($\pi = 22/7$)?

(a) 144 cm^2
(b) 154 cm^2
(c) 44 cm^2
(d) 308 cm^2

Answer: (b) — $22/7 \times 49 = 154$.

125. What is the total surface area of a cube with side 3 cm?

(a) 54 cm^2
(b) 27 cm^2
(c) 18 cm^2
(d) 36 cm^2

Answer: (a) — $6a^2 = 6 \times 9 = 54$.

126. What is the area of a triangle with sides 6, 8, 10 cm?

(a) 48 cm^2
(b) 40 cm^2
(c) 30 cm^2
(d) 24 cm^2

Answer: (d) — $6^2 + 8^2 = 10^2$, so it is right-angled; ½ × 6 × 8 = 24.

127. Two dice are thrown together. What is the probability that the sum is 8?

(a) 1/9
(b) 5/36
(c) 1/6
(d) 7/36

Answer: (b) — (2,6),(3,5),(4,4),(5,3),(6,2) = 5 outcomes.

128. What is the probability of drawing an Ace from a pack of 52 cards?

(a) 1/4
(b) 4/13
(c) 1/52
(d) 1/13

Answer: (d) — 4/52.

129. A coin is tossed 3 times. What is the probability of at least one Head?

(a) 1/8 (b) 3/8
(c) 1/2 (d) 7/8

Answer: (d) — $1 - (1/2)^3$.

130. Next term of the series: 2, 6, 12, 20, 30, ?

(a) 44 (b) 36
(c) 40 (d) 42

Answer: (d) — $n(n + 1)$: $6 \times 7 = 42$.

131. Next term of the series: 1, 2, 4, 7, 11, ?

(a) 15 (b) 16
(c) 14 (d) 17

Answer: (b) — Differences 1, 2, 3, 4, 5 $\Rightarrow$ 11 + 5 = 16 (series program, AP 2024).

132. A father's age is 3 times his son's age; after 10 years it will be 2 times. What is the son's present age?

(a) 10 (b) 15
(c) 20 (d) 12

Answer: (a) — $3S + 10 = 2(S + 10) \Rightarrow S = 10$.

133. What is the simple interest on 5000 taka at 8% for 3 years?

(a) 1300 (b) 1250
(c) 1200 (d) 1000

Answer: (c) — $5000 \times 0.08 \times 3$.

134. What is the compound interest on 10000 taka at 10% for 2 years?

(a) 2000 (b) 1900
(c) 2200 (d) 2100

Answer: (d) — $10000 \times (1.21 - 1)$.

135. What is the LCM of 12, 18 and 30?

(a) 120 (b) 90
(c) 360 (d) 180

Answer: (d) — $2^2 \times 3^2 \times 5 = 180$.

6.9 General Knowledge — Banking and ICT (136–150)

136. What is the full form of NPSB?

(a) National Postal Savings Bank (b) National Payment Switch Bangladesh
(c) National Payment System Bangladesh (d) Network Payment Service Bank

Answer: (b)

137. What is the full form of RTGS?

(a) Rapid Transfer Gross System
(b) Real Time Gross Settlement
(c) Real Transaction Global Settlement
(d) Real Time General Service

Answer: (b) — Each transaction is settled individually in real time.

138. What is the full form of BEFTN?

(a) Bangladesh E-Finance Transfer Network
(b) Bangladesh Electronic Financial Trade Network
(c) Bank Electronic Fund Transaction Node
(d) Bangladesh Electronic Funds Transfer Network

Answer: (d)

139. Where is the head office of Bangladesh Bank?

(a) Motijheel, Dhaka
(b) Agargaon, Dhaka
(c) Chattogram
(d) Gulshan, Dhaka

Answer: (a) — Motijheel commercial area, Dhaka-1000 (the circular's address also says "Motijheel, Dhaka").

140. Which was the country's first private commercial bank?

(a) Pubali Bank
(b) AB Bank
(c) Islami Bank
(d) National Bank

Answer: (b) — 1982 (then called Arab Bangladesh Bank).

141. Where was the first computer in Bangladesh installed?

(a) BCSIR
(b) BUET
(c) Atomic Energy Centre, Dhaka
(d) University of Dhaka

Answer: (c) — IBM 1620, 1964.

142. To which submarine cable was Bangladesh first connected?

(a) SEA-ME-WE 3
(b) FLAG
(c) SEA-ME-WE 5
(d) SEA-ME-WE 4

Answer: (d) — 2006, Cox's Bazar landing station.

143. In which year was Bangabandhu Satellite-1 launched?

(a) 2018
(b) 2019
(c) 2016
(d) 2017

Answer: (a) — 12 May 2018.

144. Which messaging network was misused in the 2016 Bangladesh Bank reserve heist?

(a) NPSB
(b) SWIFT
(c) BEFTN
(d) VISA

Answer: (b) — Through fake SWIFT messages against the account at the New York Fed.

145. To a bank in which country did the 81 million dollars of that reserve heist mainly go?

(a) Hong Kong
(b) Malaysia
(c) Sri Lanka
(d) Philippines

Answer: (d) — Through RCBC Bank.

146. In which country is SWIFT headquartered?

(a) Switzerland
(b) United Kingdom
(c) Belgium
(d) United States

Answer: (c) — La Hulpe, Belgium.

147. Where is the headquarters of the Bank for International Settlements (BIS)?

(a) Brussels
(b) Paris
(c) Basel
(d) Geneva

Answer: (c) — Basel, Switzerland (hence the name Basel Accords).

148. Where is the headquarters of the IMF?

(a) Washington, D.C.
(b) Geneva
(c) London
(d) New York

Answer: (a)

149. Where is the headquarters of CIRDAP?

(a) Dhaka
(b) Manila
(c) Kathmandu
(d) Bangkok

Answer: (a) — It appeared in the GK section of AP 03.01.2026.

150. What is the full form of WIPO?

(a) World International Patent Office
(b) World Information Policy Organization
(c) World Investment Promotion Organization
(d) World Intellectual Property Organization

Answer: (d) — Headquartered in Geneva; GK of AP 03.01.2026.

6.10 Answer Key (for quick checking)

1–15	16–30	31–45	46–60	61–75
1 d · 2 b · 3 b · 4 a · 5 b · 6 d · 7 c · 8 d · 9 d · 10 a · 11 b · 12 c · 13 d · 14 d · 15 a	16 d · 17 b · 18 b · 19 a · 20 b · 21 d · 22 c · 23 b · 24 c · 25 a · 26 d · 27 c · 28 c · 29 b · 30 d	31 a · 32 d · 33 b · 34 a · 35 b · 36 a · 37 a · 38 d · 39 c · 40 a · 41 a · 42 c · 43 d · 44 c · 45 a	46 c · 47 d · 48 c · 49 d · 50 d · 51 d · 52 b · 53 c · 54 a · 55 b · 56 d · 57 c · 58 d · 59 a · 60 c	61 a · 62 c · 63 a · 64 a · 65 d · 66 c · 67 d · 68 c · 69 d · 70 c · 71 b · 72 c · 73 c · 74 d · 75 c
76–90	**91–105**	**106–120**	**121–135**	**136–150**

76 b · 77 d · 78 c · 79 b · 80 d · 81 b · 82 b · 83 c · 84 c · 85 a · 86 a · 87 a · 88 c · 89 b · 90 c

91 a · 92 c · 93 a · 94 c · 95 a · 96 c · 97 b · 98 c · 99 d · 100 b · 101 a · 102 c · 103 a · 104 c · 105 d

106 c · 107 c · 108 d · 109 d · 110 b · 111 b · 112 b · 113 c · 114 d · 115 d · 116 b · 117 c · 118 d · 119 c · 120 c

121 b · 122 d · 123 d · 124 b · 125 a · 126 d · 127 b · 128 d · 129 d · 130 d · 131 b · 132 a · 133 c · 134 d · 135 d

136 b · 137 b · 138 d · 139 a · 140 b · 141 c · 142 d · 143 a · 144 b · 145 d · 146 c · 147 c · 148 a · 149 a · 150 d

Chapter 7 — Mathematics, English Language Proficiency and GK

This part carries **75 marks** in total (MCQ 25 + written 50). Everyone scores about the same in the technical part, so marks here often decide the merit list. First the solutions of all past AP questions, then practice.

7.1 Mathematics — full solutions of past AP questions

Question E · AP 17.01.2026

A rectangle and a square have the same area. The square has a perimeter of 32 m and the width of the rectangle is 4 m. Find the length of the rectangle.
Side of square = 32/4 = 8 m ⇒ area = 64 m^2
Length of rectangle = 64/4 = **16 m**

Question E · AP 17.01.2026

If $2 < x < 5$ and $3 < y < 5$, find the range of xy.
Both x and y are positive, so the bounds can be multiplied: minimum 2 × 3 = 6, maximum 5 × 5 = 25 ⇒ **$6 < xy < 25$**

Caution: if a variable can be negative, you cannot multiply directly like this — then find the minimum/maximum from the products of the four corners.

Question E · AP 17.01.2026

In a class, boys are 35% more than girls. Total students = 94. If 6 more boys are admitted, find the ratio of boys to girls.
Girls = g, boys = 1.35g ⇒ 2.35g = 94 ⇒ g = 40, boys = 54
New boys = 54 + 6 = 60 ⇒ ratio = 60 : 40 = **3 : 2**

Question E · AP 17.01.2026

A man can row 12 km/h in still water. It takes him thrice as long to row up as to row down the river. Find the rate of the stream.
Same distance, triple time ⇒ one-third speed ⇒ 12 + s = 3(12 − s)
12 + s = 36 − 3s ⇒ 4s = 24 ⇒ **speed of stream = 6 km/h** (check: downstream 18, upstream 6 — ratio 3:1 ✔)

Question B · BB AP 2023

If x is an integer and $x + 1/x = 17/4$, find $x - 1/x$.
$17/4 = 4 + 1/4$ ⇒ integer $x = 4$ ⇒ $x - 1/x = 4 - 1/4$ = **15/4**

Question B · BB AP 2023

A basketball team has won 15 games and lost 9. If these games are 16⅔% of the season, how many more games must it win to average 75%?

(The percentage is unclear in the compilation; the common version of the question has 16⅔%.)

24 = 1/6 of the total ⇒ total 144 games; 75% = 108 wins ⇒ it must win **93** more

Question B · BB AP 2023

If 4 students are extra in each row, there would be 2 rows less; if 4 are less in each row, there would be 4 more rows. Find the number of students.

$(x + 4)(y - 2) = xy \Rightarrow x = 2y - 4$; $(x - 4)(y + 4) = xy \Rightarrow x = y + 4$

$\Rightarrow y = 8$, $x = 12 \Rightarrow$ total **96 students**

The 4th question of B (area of square PSRT) depends on a figure; the compilation did not give the figure's measurements, so it is left out.

To get full marks in the 4 written maths questions

Answer structure

Define variables with "Let…" → form the equation → solve → box the answer → one-line check. There is no calculator, so work in fractions (7/20, not 0.35).

Practice set (BIBM style) — with solutions

#	Question	Solution
1	A sold to B at 20% profit; B sold to C at 25% profit. C paid 1500 taka. What is A's cost price?	1500/(1.2 × 1.25) = **1000 taka**
2	The ratio of two people's ages is 3:5; after 6 years it is 2:3. Present ages?	$(3x+6)/(5x+6) = 2/3 \Rightarrow x = 6 \Rightarrow$ **18 and 30**
3	A does a job in 12 days, B in 18 days. After working together for 4 days, A leaves. In how many days will B finish the rest?	In 4 days 4(1/12 + 1/18) = 5/9; rest 4/9 × 18 = **8 days**
4	A boat goes 15 km/h in still water, stream 3 km/h. Time to go 36 km and return?	36/18 + 36/12 = 2 + 3 = **5 hours**
5	If price rises 25%, by what % must consumption fall to keep expenditure unchanged?	25/125 × 100 = **20%**
6	Difference between compound and simple interest on 10,000 taka at 10% for 2 years?	$P(r)^2 = 10000 \times 0.01$ = **100 taka**
7	The average of 5 consecutive odd numbers is 27. The largest?	Middle one is 27 ⇒ 23, 25, 27, 29, **31**
8	If $x - y = 4$, $xy = 21$, find $x^2 + y^2$ and $x^3 - y^3$.	16 + 42 = **58**; $(x-y)^3 + 3xy(x-y)$ = 64 + 252 = **316**
9	Area of a circle is 154 cm^2 ($\pi = 22/7$). Circumference?	$r^2 = 49 \Rightarrow r = 7 \Rightarrow 2\pi r$ = **44 cm**

10	Probability that the sum of two dice is 10 or more?	(4,6),(6,4),(5,5),(5,6),(6,5),(6,6) = 6/36 = **1/6**
11	There were 45 handshakes at a meeting. How many members?	$n(n-1)/2 = 45 \Rightarrow$ **n = 10**
12	2, 6, 12, 20, 30, ?	$n(n+1) \Rightarrow 6 \times 7 =$ **42**
13	A 150 m train crosses a pole in 10 seconds. Speed in km/h?	15 m/s × 18/5 = **54 km/h**
14	Length of a rectangle increases 20%, width decreases 20%. Area?	1.2 × 0.8 = 0.96 ⇒ **decreases by 4%**
15	If $\log_x 81 = 4$, find x.	$x^4 = 81 \Rightarrow$ **x = 3**

7.2 English Language Proficiency (30 marks)

In past papers this part has come in three forms: (1) Focus + translation (A, B), (2) Focus 20 + Letter 10 (C), (3) 10 grammar items + Paragraph + Sentence making (E — the most recent AP). Preparation for all three is below.

7.2.1 Grammar — full solution of AP 17.01.2026

Sentence	Answer	Rule
A. If the department had allocated more funds, the project (would succeed / would have succeeded).	**would have succeeded**	Third conditional: If + had + V3 → would have + V3
B. By next month, the university (will have completed / will complete) the accreditation process.	**will have completed**	"By + future time" → future perfect
C. Hardly (did the presentation begin / had the presentation begun) when the technical problem occurred.	**had the presentation begun**	Hardly/Scarcely + had + subject + V3 … when
D. The committee, as well as its sub-groups, (meets / meet) regularly.	**meets**	The verb agrees with the subject before "as well as" (committee, singular)
E. He had better (leave / left) the station before it's too late.	**leave**	had better + base verb
F. The methodology section is comprehensive; (for example / however), the discussion lacks critical depth.	**however**	Opposite idea ⇒ contrast connector
G. The research team arrived at (a / an) unanimous conclusion.	**a**	"unanimous" starts with a "yoo" sound (consonant sound) ⇒ a
H. Any breach of conduct could result (in / to) suspension.	**in**	result in = lead to a consequence; result from = arise from a cause
I. She is one of the scholars who (advocates / advocate) a critical pedagogy approach.	**advocate**	"one of the + plural noun + who" ⇒ the antecedent of who is plural (scholars)

J. The researcher insisted that the data (be / is) analyzed before publication.	**be**	insist/suggest/demand + that ⇒ subjunctive (base form)

7.2.2 Ten more of the same type (with answers)

Sentence	Answer and reason
Neither the manager nor the clerks (was / were) present.	**were** — in neither...nor, the verb agrees with the nearer subject (clerks)
No sooner had he arrived (than / when) it started raining.	**than** — no sooner ... than
It is high time we (leave / left).	**left** — it is high time + past tense
He is senior (than / to) me.	**to** — senior/junior/superior/inferior + to
I look forward to (meet / meeting) you.	**meeting** — look forward to + gerund
Each of the students (has / have) a laptop.	**has** — each of + plural ⇒ singular verb
If I (was / were) you, I would accept the offer.	**were** — "were" in an unreal condition
The news (is / are) encouraging.	**is** — news is uncountable, singular
She prevented him (from / to) going.	**from** — prevent somebody from + gerund
Scarcely had we reached the platform (when / than) the train left.	**when** — scarcely/hardly ... when

7.2.3 Sentence making E · AP 17.01.2026

Word (part of speech)	Sentence
Judicious (adjective)	A judicious use of public funds can speed up rural development.
Condone (verb)	The bank will not condone any breach of its information security policy.
Fabricate (verb)	The borrower fabricated documents to obtain the loan.
Accentuate (verb)	The new interface accentuates the most important account information.
Multifaceted (adjective)	Cyber security is a multifaceted challenge for modern banks.

Remember the rule

"Do not change the grammatical category" — if you turn an adjective into an adverb (judiciously) or a verb into a noun (fabrication), you get no marks.

7.2.4 Paragraph — "Street Food Culture in Dhaka" E · AP 17.01.2026

Model (~170 words)

Street food is an inseparable part of Dhaka's everyday life. From fuchka and chotpoti near the university areas to jhalmuri on the footpaths and kebabs in Old Dhaka, the city offers cheap, tasty food at almost every corner. For students, office workers and rickshaw pullers alike, these stalls provide quick meals at affordable prices, and during Ramadan the iftar markets of Chawkbazar become a cultural festival in themselves. Street food also supports thousands of small vendors and their families, making it an important part of the informal economy. However, the culture faces serious challenges. Many stalls use unsafe water, reused oil and uncovered food, which can cause diarrhoea, typhoid and other diseases. Vendors are rarely trained in hygiene, and there is little monitoring by the authorities. If the city corporations introduce licensing, training and designated vending zones with clean water, Dhaka's street food can become both safe and a major attraction for tourists. With proper care, this colourful tradition can continue to feed the city while protecting public health.

7.2.5 Letter (10 marks in Paper C)

Structure of a formal letter

Date → Recipient (designation, organisation, address) → Subject (one line) → Salutation (Dear Sir/Madam,) → Body (3 paragraphs: purpose · details · request/action) → Closing (Yours faithfully,) → Name, designation, contact.

Sample: complaint about an unauthorised transaction in online banking

31 October 2026
The Branch Manager
ABC Bank PLC, Motijheel Branch, Dhaka-1000

Subject: Complaint regarding an unauthorised internet banking transaction

Dear Sir/Madam,
I am writing to report an unauthorised transfer of Tk 25,000 from my savings account (No. 0123456789) through internet banking on 29 October 2026 at 10:42 pm. I did not initiate this transaction, nor did I share my password or OTP with anyone.
I immediately blocked my internet banking access through the call centre (reference no. CC-55871). However, the amount has not yet been reversed, and I am concerned about the security of my account.
I therefore request you to investigate the transaction, reverse the amount if it is found to be fraudulent, and advise me on any further steps I should take. Copies of my account statement and the SMS alert are attached.

Yours faithfully,
(Signature)
Name · Account No. · Mobile

7.2.6 Focus / Essay — structure and model

Model: Online Banking D · BSCS 03.01.2026

Introduction. Online banking allows customers to check balances, transfer funds, pay bills and open deposits through the internet or a mobile app, without visiting a branch.
Bangladesh context. Banks now offer internet and app banking alongside interoperable channels such as NPSB, BEFTN, RTGS and Bangla QR, while mobile financial services have brought millions of unbanked people into the digital system.
Benefits. It saves time and transaction cost, operates 24/7, reduces cash handling and branch queues, and gives banks better data for credit and risk decisions.
Challenges. Phishing, OTP fraud and weak passwords; limited digital literacy, especially among rural and elderly customers; uneven internet coverage; and the need for strong cyber-security infrastructure in banks.
Way forward. Multi-factor authentication, real-time fraud monitoring, customer awareness campaigns, simple Bangla-language interfaces and strict compliance with Bangladesh Bank's ICT security guideline.
Conclusion. With trust and security in place, online banking can be the backbone of a cashless, inclusive Smart Bangladesh.

Bangla essay: "The Role of Youth in Building a Corruption-Free Country" D **— outline**

Introduction (what corruption is, a barrier to development) → forms and harms of corruption → why youth matter (large share of population, tech-skilled) → role: staying honest oneself, awareness and volunteering, using the Right to Information Act and digital services, demanding accountability on social media, honesty in leadership → obstacles → recommendations → conclusion.

7.2.7 Translation — past AP questions

Bangla → English D · BSCS 03.01.2026

বর্তমান ডিজিটাল যুগে গুগল ড্রাইভ এখন শুধু একটি ফাইল সংরক্ষণের জায়গা নয়, বরং এটি পড়াশোনা, অফিসের কাজ ও ব্যক্তিগত নথি ব্যবস্থাপনার একটি গুরুত্বপূর্ণ ডিজিটাল মাধ্যম। প্রতিদিনই কোটি কোটি মানুষ এ ক্লাউড স্টোরেজ সেবার ওপর নির্ভর করছে। তবে নিয়মিত ব্যবহার করলেও অনেকেই জানেন না, গুগল ড্রাইভে এমন কিছু লুকানো ফিচার রয়েছে, যা ব্যবহারকারীর কাজকে আরও সহজ, দ্রুততর ও কার্যকর করে তুলতে পারে। ই-মেইলে বড় ফাইল পাঠানোর ক্ষেত্রে অ্যাটাচমেন্ট সাইজ একটি বড় সমস্যা। তবে জিমেইল ব্যবহারকারীরা গুগল ড্রাইভ থেকে সরাসরি ফাইল যুক্ত করে সহজেই বড় ফাইল পাঠাতে পারেন।

Translation: In today's digital age, Google Drive is no longer just a place to store files; it has become an important digital medium for studies, office work and the management of personal documents. Every day, hundreds of millions of people rely on this cloud storage service. Yet, even though they use it regularly, many are unaware that Google Drive has some hidden features that can make their work easier, faster and more effective. Attachment size is a major problem when sending large files by e-mail. However, Gmail users can easily send large files by attaching them directly from Google Drive.

English → Bangla D · BSCS 03.01.2026

From the Aman paddy season to the current winter vegetable cultivation, farmers in many districts have been unable to purchase fertilizers at government-fixed prices. Farmers and government officials allege that dishonest traders are exploiting an "artificial crisis", particularly for non-urea fertilizers, making extra profits. Potato farmers in northern districts are among the worst affected. In many districts, farmers reported paying Tk 200–600 more per 50 kg sack than the government rates for non-urea fertilizers. In response, local administration seized around 12,000 sacks of illegally stockpiled fertilizer.

Translation (Bangla answer): আমন মৌসুম থেকে শুরু করে চলতি শীতকালীন সবজি চাষ পর্যন্ত অনেক জেলার কৃষকেরা সরকার-নির্ধারিত দামে সার কিনতে পারছেন না। কৃষক ও সরকারি কর্মকর্তাদের অভিযোগ, অসাধু ব্যবসায়ীরা বিশেষ করে নন-ইউরিয়া সারের "কৃত্রিম সংকট" তৈরি করে বাড়তি মুনাফা করছেন। উত্তরাঞ্চলের জেলাগুলোর আলুচাষিরা সবচেয়ে বেশি ক্ষতিগ্রস্ত। অনেক জেলায় কৃষকেরা জানিয়েছেন, নন-ইউরিয়া সারের প্রতি ৫০ কেজির বস্তায় তাঁদের সরকারি দামের চেয়ে ২০০ থেকে ৬০০ টাকা বেশি দিতে হচ্ছে। এর প্রেক্ষিতে স্থানীয় প্রশাসন অবৈধভাবে মজুত করা প্রায় ১২ হাজার বস্তা সার জব্দ করেছে।

English → Bangla (partial) B · BB AP 2023

Global economic turbulence is strongly felt in the banking and financial industry throughout the globe. The visible transformations are taking place in terms of embracing new recovery efforts by adopting newer technology...

Translation (Bangla answer): বৈশ্বিক অর্থনৈতিক অস্থিরতা সারা বিশ্বের ব্যাংকিং ও আর্থিক খাতে তীব্রভাবে অনুভূত হচ্ছে। নতুন প্রযুক্তি গ্রহণের মাধ্যমে পুনরুদ্ধারের নতুন উদ্যোগ নেওয়ার ক্ষেত্রে দৃশ্যমান পরিবর্তন ঘটছে...

Banking and technology vocabulary (Bangla → English)

ঋণ খেলাপি = loan defaulter · মওকুফ = waive · মজুত = stockpile · জব্দ = seize · আস্থা = confidence/trust · জবাবদিহিতা = accountability · আর্থিক অন্তর্ভুক্তি = financial inclusion · লেনদেন = transaction · নগদবিহীন = cashless · তারল্য সংকট = liquidity crisis · মূল্যস্ফীতি = inflation · প্রণোদনা = incentive · উদ্যোক্তা = entrepreneur

7.3 General Knowledge

Past AP questions D · BSCS 03.01.2026

Question	Answer
Where is the first earthquake observatory in Bangladesh?	Chattogram (1954)
Full form of WIPO	World Intellectual Property Organization
When was VAT introduced in Bangladesh?	1991 (1 July)
First bank owned by Bengali entrepreneurs	Eastern Mercantile Bank (1959; now Pubali Bank) — the common answer; some sources also mention Comilla Banking Corporation of the British period
Headquarters of CIRDAP	Dhaka
Inventor of dynamite	Alfred Nobel
Which country won the men's singles at the ITTF World Championships 2025?	China (Wang Chuqin)
How many layers are in the OSI model?	7
Meaning of "Terracotta"	Baked clay (Italian: baked earth)
Which organisation grants GSP+ facility?	European Union (EU)

Banking and ICT — quick revision

Question	Answer
NPSB	National Payment Switch Bangladesh — interconnects ATM/POS/online transactions of different banks

BEFTN / RTGS / BACH	Bangladesh Electronic Funds Transfer Network (batch) / Real Time Gross Settlement (instant, single) / Bangladesh Automated Clearing House
Governor of Bangladesh Bank, headquarters	Headquarters in Motijheel, Dhaka (check the name of the latest governor in the newspaper before the exam)
First private commercial bank	AB Bank (1982)
First computer in Bangladesh	IBM 1620, Atomic Energy Centre, Dhaka (1964)
First submarine cable	SEA-ME-WE 4 (2006, Cox's Bazar); second SEA-ME-WE 5 (Kuakata)
Bangabandhu Satellite-1	12 May 2018
IMF / World Bank headquarters	Washington D.C.
SAARC / BIMSTEC headquarters	Kathmandu / Dhaka
International Literacy Day	8 September

Current affairs

The GK part of the MCQ (10) may include recent events (for example ITTF 2025 in D). The day before the exam, spend 15 minutes on the international and economy pages of a newspaper — recent figures/names are deliberately kept few in this book.

Chapter 8 — Model Tests and Last-Night Checklist

> **How to take it**
>
> Start at 9:30 AM, like the real exam. MCQ for exactly 60 minutes, then written for 120 minutes — no break in between. Keep the phone away, no calculator. At the end, score the MCQ as (correct − 0.25 × wrong); check the written answers against the solutions in the chapter indicated.

8.1 Model Test 1

Part A — MCQ (100 marks, 60 minutes)

Do the following 100 questions from Chapter 6 in one go (keep the answers covered): **Questions 1–75** (Subjects Related 75) + **Questions 111–125** (Mathematics 15) + **Questions 136–145** (GK 10)ı

Part B — Written (200 marks, 120 minutes)

Subjects Related (15 × 10 = 150) — the brackets show where to find the answer

1. Insert the keys 12, 44, 13, 88, 23, 94, 11, 39, 20 into a hash table of size 11 using h(k) = k mod 11 and linear probing. Show the final table. (Chapter 4 — hashing)
2. Write a C or Java program that prints all prime numbers between 1 and n, where n is given by the user. (Chapter 3)
3. Write a program to compute the GCD and LCM of two integers. (Chapter 3)
4. Given EMPLOYEE(EmpID, Name, DeptID, Salary) and DEPARTMENT(DeptID, DeptName), write SQL to (a) list employees earning more than their department's average, (b) show each department name with its employee count, sorted by count descending. (Chapter 4)
5. Draw a UML class diagram: a Bank has many Branches; a Branch serves many Customers; a Customer may hold several Accounts; SavingsAccount and CurrentAccount are kinds of Account. (Chapter 1 — class diagram rules)
6. Differentiate method overloading and method overriding with a Java example. When does method hiding occur? (Chapter 1 and 5)
7. Show the max-heap after inserting 15, 7, 20, 3, 18, 25 one by one, then after one extract-max. (Chapter 4)
8. Explain ACID properties with the example of a fund transfer between two accounts. (Chapter 5)
9. What is a deadlock? State the four necessary conditions and one prevention technique for each. (Chapter 5)
10. Compare TCP and UDP. Explain the TCP three-way handshake. (Chapter 2 and 5)
11. Explain the CIA triad. Differentiate hashing and encryption with one algorithm example each. (Chapter 2 and 5)
12. Critically compare the Waterfall model and Agile/Scrum. Which would you choose for a mobile banking app and why? (Chapter 2 and 5)
13. Count page faults for the reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 with 3 frames using FIFO and LRU. (Chapter 4)

14. Prove that NAND is a universal gate. (Chapter 2)
15. Explain how a queue can be implemented using two stacks. Give the time complexity of each operation. (Chapter 2 and 4)

Mathematics (4 × 5 = 20)

1. A shopkeeper marks goods 40% above cost and allows a 15% discount. Find the profit percentage.
2. The sum of three consecutive even numbers is 78. Find the numbers.
3. Pipe A fills a tank in 6 hours and pipe B empties it in 9 hours. If both are opened together, when will the tank be full?
4. A boat's upstream speed is 8 km/h and downstream speed is 14 km/h. Find the speed in still water and of the stream.

English Language Proficiency (30)

1. Choose the correct option (10): (a) Neither of the answers (is/are) correct. (b) Had I known, I (would help/would have helped) you. (c) She has been working here (since/for) 2020. (d) The jury (was/were) divided in their opinions. (e) He is accustomed (to work/to working) late. (f) Not only the students but also the teacher (was/were) late. (g) It is an (unique/a unique) opportunity — choose the correct article phrase. (h) The manager suggested that he (submit/submits) the report. (i) Rarely (we see/do we see) such dedication. (j) The bank is (comprised of/composed of) five divisions.
2. Write a paragraph (≈150 words) on "Cyber Security Awareness for Bank Customers". (10)
3. Make sentences without changing the part of speech (10): Prudent, Mitigate, Streamline, Robust, Scrutinize.

Answers — Model Test 1

Part	Answer
IT 1 (hash, m = 11)	12→1, 44→0, 13→2, 88→0→1→2→3, 23→1→…→4, 94→6, 11→0→…→5, 39→6→7, 20→9 ⇒ index 0:44, 1:12, 2:13, 3:88, 4:23, 5:11, 6:94, 7:39, 9:20
IT 7 (heap)	After all inserts [25, 18, 20, 3, 7, 15]; extract-max returns 25 ⇒ [20, 18, 15, 3, 7]
IT 13 (paging)	FIFO = 10 faults, LRU = 9 faults (build the step table yourself using the Chapter 4 method and verify)
Math 1	1.40 × 0.85 = 1.19 ⇒ **19% profit**
Math 2	x + (x+2) + (x+4) = 78 ⇒ x = 24 ⇒ **24, 26, 28**
Math 3	1/6 − 1/9 = 1/18 ⇒ **18 hours**
Math 4	Still water (8 + 14)/2 = **11 km/h**; stream (14 − 8)/2 = **3 km/h**
English 1	(a) is (b) would have helped (c) since (d) were (members have separate opinions — plural) (e) to working (f) was (g) a unique (h) submit (i) do we see (j) composed of

8.2 Model Test 2

Part A — MCQ (100 marks, 60 minutes)

From Chapter 6: **Questions 36–110** (Subjects Related 75) + **Questions 121–135** (Mathematics 15) + **Questions 141–150** (GK 10). Some questions will overlap with Model Test 1 — the point is to see whether you have fixed your earlier mistakes on them.

Part B — Written (200 marks, 120 minutes)

Subjects Related (15 × 10 = 150)

1. Write a program to print Floyd's triangle (numbers) for n rows, and a second version that prints the 0/1 pattern. (Chapter 1 and 3)
2. Write a function that returns the second largest element of an array in one pass. (Chapter 3)
3. Trace the output of a recursive function that prints before and after its recursive call (e.g. f(n): if n ≤ 0 return; print n; f(n−2); print n) for f(5). (Chapter 1 and 3)
4. Write SQL to find the second highest salary, and to find movies that belong to both 'Romance' and 'Comedy' genres. (Chapter 1 and 4)
5. Explain normalization up to 3NF with an example table. (Chapter 5)
6. Represent a given directed graph using an adjacency list and an adjacency matrix. Which is better for a sparse graph, and why? (Chapter 1 and 4)
7. Run Dijkstra's algorithm on a 5-node weighted graph and show the distance table after each step. (Chapter 4)
8. Describe the four pillars of OOP with a short Java example of each. (Chapter 5)
9. Explain the producer–consumer problem and its solution with semaphores. (Chapter 2 and 5)
10. Describe a man-in-the-middle attack on Diffie–Hellman key exchange and how authentication prevents it. (Chapter 1)
11. A new employee joins the bank. Describe secure user-account provisioning (password policy, MFA, least privilege, review, offboarding). (Chapter 2 and 5)
12. Differentiate a process and a thread; give two advantages of multithreading. (Chapter 2 and 5)
13. Explain IaaS, PaaS and SaaS with examples; differentiate a virtual machine and a container. (Chapter 2 and 5)
14. Explain how OSPF builds its routing table. (Chapter 2 and 5)
15. Compare supervised, unsupervised and reinforcement learning with one banking example each. (Chapter 2 and 5)

Mathematics (4 × 5 = 20)

1. If $x + y = 10$ and $x - y = 4$, find xy and $x^2 + y^2$.
2. 30% of a number is 45 more than 15% of it. Find the number.
3. A bag has 5 red and 3 blue balls. Two balls are drawn without replacement. Find the probability that both are red.
4. The diagonal of a square is $10\sqrt{2}$ cm. Find its area and perimeter.

English Language Proficiency (30)

1. Correct the sentences (10): (a) He is one of the best player of the team. (b) I have seen him yesterday. (c) Each of the officers have a laptop. (d) She is more cleverer than her sister. (e) The furnitures are new. (f) We discussed about the problem. (g) He did not came to office. (h) Unless you do not work hard, you will fail. (i) I prefer tea than coffee. (j) The customer has been waiting since two hours.
2. Write a formal letter to the Branch Manager requesting activation of internet banking for your account. (10)
3. Write a paragraph on "Digital Literacy and Cashless Bangladesh". (10)

Answers — Model Test 2

Part	Answer
IT 3 (f(5))	f(5): prints 5 → f(3): 3 → f(1): 1 → f(−1) returns → prints 1 → 3 → 5 ⇒ **5 3 1 1 3 5**
Math 1	xy = (100 − 16)/4 = **21**; $x^2 + y^2 = 100 - 42 =$ **58**
Math 2	0.15n = 45 ⇒ **n = 300**
Math 3	(5/8)(4/7) = 20/56 = **5/14**
Math 4	Side = 10 cm ⇒ area **100 cm²**, perimeter **40 cm**
English 1	(a) best players (b) I saw him yesterday (c) Each of the officers has (d) cleverer (e) The furniture is new (f) discussed the problem (g) did not come (h) Unless you work hard (i) prefer tea to coffee (j) for two hours

8.3 Last-Night Checklist (Friday night)

✓	Item
☐	Print the admit card (1 copy) — write nothing on it; carry only a pen
☐	Route and travel time to Tejgaon Government High School fixed; reach by 8:30 (entry closes after 9:15)
☐	Phone, watch, calculator, bag, bank card — none of them
☐	Review one last time: hash/heap/LRU trace method · SQL JOIN + GROUP BY + HAVING · overloading/overriding/hiding · OSI + ports · ACID · deadlock 4 conditions · UML class diagram symbols (inheritance ▷, composition ◆, aggregation ◇)
☐	English: conditionals, subjunctive (insist that + base), "one of the … who + plural verb", hardly/scarcely … when, no sooner … than
☐	Go through your own mistake list once — then sleep

> **Remember in the hall**
>
> Do not get stuck on a hard question — move to the next. In every written answer give at least a definition + table/diagram + example — you get partial marks even if you do not know it all. In MCQ, if you cannot eliminate even one option, leave it blank.