ব্যাংকার্স সিলেকশন কমিটি সচিবালয় (BSCS) · Job ID-25106

Assistant Programmer
বিস্তারিত প্রস্তুতি

Assistant Programmer / Senior Officer (AP) · ৯ম গ্রেড · ৩৩টি পদ

৬টি বিগত AP প্রশ্নের পূর্ণ সমাধান (৩১.১০.২০২৬ সহ) | প্রোগ্রামিং প্রবলেম ব্যাংক | SQL ও DSA হাতে-কলমে | রেডি থিওরি উত্তর | ১৫০ MCQ | গণিত · English · GK | মডেল টেস্ট

পরীক্ষা: ৩১ অক্টোবর ২০২৬, শনিবার · সকাল ৯:৩০ – ১২:৩০ · তেজগাঁও সরকারি উচ্চ বিদ্যালয়, ঢাকা

সূচিপত্র ও পরীক্ষা বিশ্লেষণ

Contents, Exam Facts & Past-Paper Analysis

অধ্যায়	বিষয়	কেন	পৃষ্ঠা
০	পরীক্ষা তথ্য, বিগত AP প্রশ্ন বিশ্লেষণ, ৩ দিনের রুটিন	কী আসে, কোনটা আগে	২
১	সমাধানকৃত বিগত প্রশ্ন: Combined 5 Bank ২০২৩, Bangladesh Bank ২০২৩, Combined 5 Bank ২০২৪	BIBM-এর সমস্যাভিত্তিক প্রশ্ন	৬
২	সমাধানকৃত বিগত প্রশ্ন: BSCS ০৩.০১.২০২৬, BSCS ১৭.০১.২০২৬, BSCS ২০২০, **BSCS ৩১.১০.২০২৬ (Job ID-25106, সম্পূর্ণ)**, **বাছাই MCQ ২০২৬ সেট-৩ (১০০টি MCQ)**	সাম্প্রতিক থিওরিভিত্তিক প্রশ্ন	২৯
৩	প্রোগ্রামিং প্রবলেম ব্যাংক + Output tracing	প্রতি পেপারে ২–৩টি কোড	৯৯
৪	SQL কুয়েরি ব্যাংক ও DSA হাতে-কলমে (heap, hash, graph, sort, paging)	প্রতি পেপারে SQL; প্রায় সবটায় DSA	১২৫
৫	১০ নম্বরের থিওরি: রেডি উত্তর	২০২৬-এর পেপার বেশিরভাগ "Explain..."	১৪৫
৬	AP MCQ ব্যাংক (১৫০টি)	MCQ ১০০ নম্বর, −০.২৫	১৬৭
৭	গণিত, English Language Proficiency ও GK	লিখিতে ৫০, MCQ-তে ২৫ নম্বর	১৮৬
৮	মডেল টেস্ট (২ সেট) ও শেষ রাতের চেকলিস্ট	সময় ধরে অনুশীলন	১৯৪

০.১ এই পরীক্ষা সম্পর্কে যা নিশ্চিত (বিজ্ঞপ্তি নং ৩৪০/২০২৬)

বিষয়	তথ্য
পদ	Assistant Programmer / Senior Officer (Assistant Programmer), ৯ম গ্রেড, Job ID-25106 — BSCS-এর ৫টি ব্যাংক ও ২টি আর্থিক প্রতিষ্ঠানে ৩৩টি পদ
পরীক্ষার্থী	২৪৮২ জন (রোল ৬০০০১–৬৩৩১০) — প্রতি পদে ~৭৫ জন
সময়	৩১/১০/২০২৬ শনিবার, সকাল ৯:৩০ – ১২:৩০ (MCQ ১ ঘণ্টা + লিখিত ২ ঘণ্টা, একই বসায়)
কেন্দ্র	তেজগাঁও সরকারি উচ্চ বিদ্যালয়, তেজগাঁও, ঢাকা; **৯:১৫-এর পর প্রবেশ নিষেধ**
MCQ (১০০)	Subjects Related ৭৫ · Mathematics ১৫ · General Knowledge ১০ · প্রতি ভুলে −০.২৫
লিখিত (২০০)	Subjects Related ১৫০ · Mathematics ২০ · English Language Proficiency ৩০
সঙ্গে নেওয়া যাবে	শুধু প্রবেশপত্র (১ কপি; erecruitment.bb.org.bd থেকে ডুপ্লিকেট নামানো যায়)। ফোন, ক্যালকুলেটর, ঘড়ি, ব্যাগ, কার্ড, কাগজ নিষিদ্ধ। দুই কান দৃশ্যমান রাখতে হবে।

যা নিশ্চিত নয়: প্রশ্নকর্তা

বিজ্ঞপ্তিতে নাম নেই। ২০২৩–২০২৬ সালে "১ ঘণ্টা MCQ + ২ ঘণ্টা লিখিত একই বসায়" ফরম্যাটের সব ব্যাংক-আইটি পরীক্ষা সংকলনে BIBM-এর নামে আছে (সর্বশেষ AP ১৭.০১.২০২৬: "BIBM/CTI")। তাই **BIBM সবচেয়ে সম্ভাব্য** — এটি অনুমান। বিগত প্রশ্নগুলো পরীক্ষার্থীদের স্মৃতি থেকে সংকলিত, তাই ভাষা হুবহু নাও হতে পারে।

০.২ বিগত AP প্রশ্নপত্র — এক নজরে

সংকেত	পরীক্ষা	প্রশ্নকর্তা	লিখিত কাঠামো
A	Combined 5 Bank AP, ০৩.০২.২০২৩	BIBM	MCQ ৮০ (IT ৭৫ + Math/GK ২৫); লিখিত IT ১৫০ + Math ২০ + Focus ১০ + অনুবাদ ২০
B	Bangladesh Bank AP, ০৩.০২.২০২৩	BIBM	৯টি IT প্রশ্ন (১৫০) + Math ৪টি (২০) + Focus ১০ + অনুবাদ ২০
C	Combined 5 Bank (10198) AP, ০৯.০২.২০২৪	BIBM	১০টি IT প্রশ্ন (১৫০) + Math ৪টি (২০) + Focus ২০ + Letter ১০
D	BSCS AP / Officer (IT), ০৩.০১.২০২৬	—	Non-tech (Bangla essay, English essay, অনুবাদ ২টি, GK ১০টি) + ১৫টি IT প্রশ্ন × ১০
E	BSCS Various Bank AP, ১৭.০১.২০২৬	BIBM/CTI	১৫টি IT প্রশ্ন × ১০ + Math ৪টি + English (grammar ১০টি, paragraph, sentence making)
F	BSCS AP (Job ID-25106), ৩১.১০.২০২৬ — **এই বিজ্ঞপ্তির পরীক্ষা**	—	১৫টি IT প্রশ্ন × ১০ (৪টি আধুনিক AI) + Math ৪টি × ৫ + English ৩০ (grammar ১০টি + ২০ নম্বরের essay) — অনুচ্ছেদ ২.৪-এ সম্পূর্ণ সমাধান
G	বাছাই (প্রিলিমিনারি) MCQ পরীক্ষা ২০২৬, সেট-৩ (হেডারে সম্ভবত পায়রাবন্দর)	—	শুধু MCQ: ১০০টি প্রশ্ন, ১ ঘণ্টা, ভুলে −০.৫০ — IT ৩৩, Electrical ৭, English ২০, বাংলা ২০, GK ১৯, Math ১; অনুচ্ছেদ ২.৫-এ সম্পূর্ণ সমাধান

E-এর English অংশই এবারের বিজ্ঞপ্তির "English Language Proficiency ৩০"-এর সবচেয়ে কাছের নমুনা — অধ্যায় ৭-এ পুরো সমাধান আছে।

০.৩ টপিক কতবার এসেছে (৬টি AP পেপার)

টপিক	A	B	C	D	E	F	উদাহরণ
DBMS / SQL	✔	✔	✔	✔	✔	✔	M:N join কুয়েরি, 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 traversal
প্রোগ্রাম লেখা	✔	✔	✔	—	✔	—	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

চারটি বড় শিক্ষা

1. **SQL ও Software Engineering প্রতিটি AP পেপারে এসেছে** — অধ্যায় ৪-এর SQL অংশ ও UML class diagram বাদ দেওয়ার সুযোগ নেই।
2. **ধরন বদলেছে:** ২০২৩–২৪-এ প্রশ্ন ছিল হিসাব/কোডভিত্তিক (heap trace, hash table, প্রোগ্রাম); ২০২৬-এ প্রায় সব "Explain / Differentiate" (১৫টি × ১০)। তাই অধ্যায় ৩–৪ (হাতে-কলমে) *এবং* অধ্যায় ৫ (রেডি থিওরি) দুটোই লাগবে।
3. **AI এখন মূল টপিক:** Paper F (৩১.১০.২০২৬)-এ AI নিয়ে ৪টি প্রশ্ন ছিল — NLP, Generative AI/LLM, MCP-সহ agentic AI এবং RAG — সাথে server capacity ও cache AMAT-এর হিসাব। অনুচ্ছেদ ২.৪ এবং অধ্যায় ৫-এর Cloud/AI অংশ একসাথে রিভিশন দিন।
4. **ব্যাংকিং প্রেক্ষাপট:** "new employee account", "permission precedence", "RAID for bank storage" — উত্তরে ব্যাংকের উদাহরণ দিন।

০.৪ ৩ দিনের রুটিন (এই বই অনুযায়ী)

দিন	সময়	কী করবেন
দিন ১ — বুধ, ২৮ অক্টো	প্রথম ভাগ	অধ্যায় ১ পুরোটা (A, B, C) — প্রতিটি প্রশ্ন আগে নিজে চেষ্টা, তারপর সমাধান মেলান
	দ্বিতীয় ভাগ	অধ্যায় ৩: প্রথম ১৫টি প্রোগ্রাম কাগজে লিখুন + output tracing অংশ
	রাত	অধ্যায় ৬: Programming, DSA ও DBMS অংশের MCQ — ভুলের তালিকা বানান
দিন ২ — বৃহস্পতি, ২৯ অক্টো	সকাল	অধ্যায় ২ (D, E, 2020) — বিশেষ করে E, কারণ এটি সবচেয়ে সাম্প্রতিক AP
	দুপুর–বিকেল	অধ্যায় ৪: SQL-এর ৩০টি কুয়েরি (হাতে) + heap/hash/sort/paging trace
	রাত	অধ্যায় ৫: OOP, DBMS, OS, Security অংশ; অধ্যায় ৬-এর বাকি MCQ
দিন ৩ — শুক্র, ৩০ অক্টো	সকাল	অধ্যায় ৫-এর বাকি (Networking, SE, Cloud/AI, Digital) + অধ্যায় ৭ (গণিত, English, GK)
	৯:৩০–১২:৩০	**মডেল টেস্ট ১ (অধ্যায় ৮)** — আসল পরীক্ষার সময়েই, ঘড়ি ধরে; তারপর নম্বর দিন (−০.২৫ সহ)
	বিকেল–রাত	মডেল টেস্ট ২-এর লিখিত অংশ (শুধু আউটলাইন) + ভুলের তালিকা + অধ্যায় ৮-এর শেষ রাতের চেকলিস্ট; ১১টার মধ্যে ঘুম
শনি, ৩১ অক্টো	সকাল	নতুন কিছু নয়। ৮:৩০-এর মধ্যে কেন্দ্রে। শুধু প্রবেশপত্র সঙ্গে।

০.৫ হলে সময় ও নম্বর কৌশল

অংশ	সময়	কৌশল
MCQ ১০০	৬০ মিনিট	রাউন্ড ১: নিশ্চিতগুলো (~৩৫ মিনিট); রাউন্ড ২: একটি অপশন বাদ দিতে পারলে উত্তর দিন, না পারলে ছাড়ুন; শেষ ৫ মিনিট OMR মেলান
লিখিত IT ১৫০	~৯০ মিনিট	১৫টি প্রশ্ন হলে ~৬ মিনিট করে; জানা প্রশ্ন আগে; "Differentiate" হলে ৫–৬ সারির টেবিল; কোডে সংক্ষিপ্ত কমেন্ট ও sample I/O
গণিত ২০	~১২ মিনিট	ধাপ দেখান — ভুল উত্তরেও ধাপে নম্বর আসে
English ৩০	~১৮ মিনিট	grammar আগে (দ্রুত নম্বর), তারপর paragraph/letter; ছোট, নির্ভুল বাক্য

−০.২৫ নেগেটিভ মার্কিং: অঙ্কে দেখা

অন্ধ আন্দাজ (৪টি অপশন): ¼ − ¾ × 0.25 = +০.০৬ · একটি বাদ দিতে পারলে: ⅓ − ⅔ × 0.25 = +০.১৭ · দুটি বাদ দিতে পারলে: ½ − ½ × 0.25 = +০.৩৭৫। **সিদ্ধান্ত:** অন্তত একটি অপশন নিশ্চিতভাবে বাদ দিতে পারলে উত্তর দিন।

অধ্যায় ১ — সমাধানকৃত বিগত প্রশ্ন: ২০২৩ ও ২০২৪

Solved AP Papers — Combined 5 Bank 2023, Bangladesh Bank 2023, Combined 5 Bank 2024 (all BIBM)

BIBM-এর তিনটি Assistant Programmer লিখিত পরীক্ষার **সব টেকনিক্যাল প্রশ্নের** পূর্ণ মডেল উত্তর এখানে দেওয়া হলো। গণিত, ইংরেজি, অনুবাদ ও Focus Writing-এর সমাধান আলাদা অধ্যায়ে আছে। প্রশ্নগুলো পরীক্ষার্থীদের স্মৃতি থেকে সংকলিত, তাই কিছু জায়গায় ভাষা অসম্পূর্ণ ছিল। সেসব জায়গায় যে ব্যাখ্যা ধরে নেওয়া হয়েছে, তা উত্তরেই স্পষ্ট লেখা আছে।

কীভাবে ব্যবহার করবেন

প্রথমে শুধু প্রশ্নের বক্সটি পড়ুন এবং ৬–৮ মিনিটে খাতায় নিজে উত্তর লিখুন। তারপর মডেল উত্তরের সঙ্গে মেলান। Heap, hashing ও LRU-র ধাপের টেবিল এবং কোড **হাতে লিখে** অনুশীলন করুন।

১.১ Combined 5 Bank — Assistant Programmer (০৩.০২.২০২৩)

তথ্য	বিবরণ
তারিখ	০৩.০২.২০২৩
প্রশ্নকর্তা	BIBM
ধরন	৮০ MCQ (IT ৭৫ + গণিত ও GK ২৫); লিখিত ২০০ = IT ১৫০ + গণিত ৪টি (২০) + Focus ১০ + অনুবাদ B/E ২০
টপিক	Heap, SQL join, UML class diagram, Function Point, array function, TLS, CRC, graph representation, browser security, LRU

প্রশ্ন ১ Combined 5 Bank AP 2023

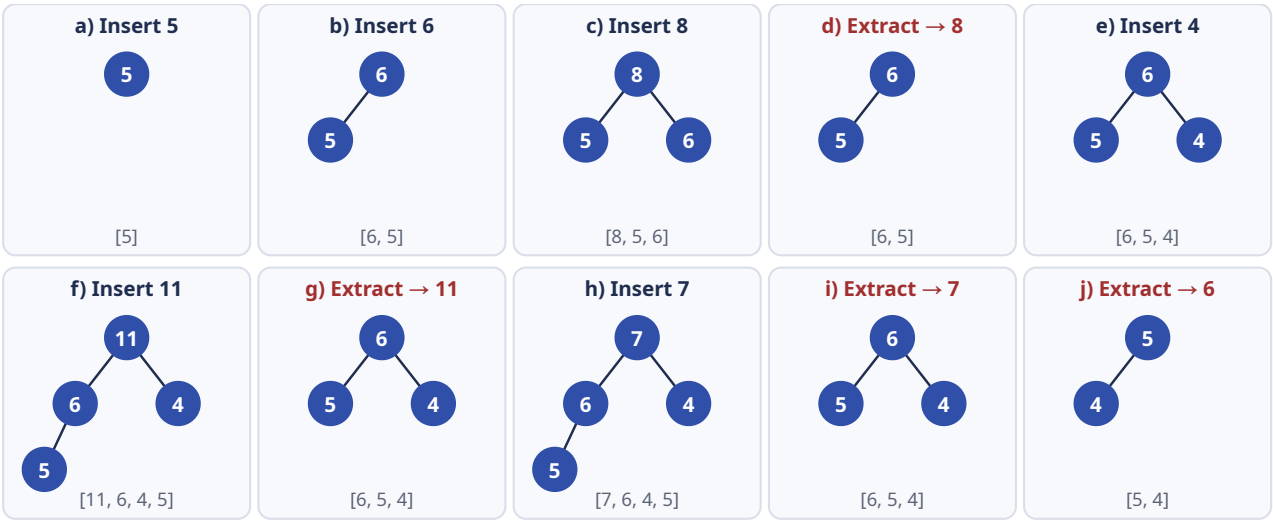
Show the Max Heap after each of the following operations [a–j]. Also show which value is returned whenever the root is extracted: Insert 5, Insert 6, Insert 8, Extract-Root, Insert 4, Insert 11, Extract-Root, Insert 7, Extract-Root, Extract-Root.

নিয়ম মনে রাখুন

Insert: নতুন মান array-এর শেষে বসান। তারপর **sift-up** করুন, অর্থাৎ parent (index $\lfloor (i-1)/2 \rfloor$) ছোট হলে swap। **Extract-Root:** root (সর্বোচ্চ মান) ফেরত দিন। শেষ উপাদানটি root-এ এনে **sift-down** করুন, অর্থাৎ বড় child-এর সঙ্গে swap (children: 2i+1, 2i+2)।

ধাপ	অপারেশন	কী ঘটে	Heap (array)	ফেরত
a	Insert 5	খালি heap, 5 root হয়	[5]	—
b	Insert 6	6 শেষে বসে; parent 5 < 6, তাই swap	[6, 5]	—
c	Insert 8	index 2-এ বসে; parent 6 < 8, তাই swap	[8, 5, 6]	—
d	Extract-Root	8 বের হয়; শেষের 6 root-এ আসে; child 5 < 6, তাই কিছু বদলায় না	[6, 5]	**8**
e	Insert 4	index 2-এ বসে; parent 6 > 4, তাই ঠিক আছে	[6, 5, 4]	—
f	Insert 11	index 3-এ বসে; parent 5-এর সঙ্গে swap, তারপর root 6-এর সঙ্গে swap	[11, 6, 4, 5]	—
g	Extract-Root	11 বের হয়; 5 root-এ আসে; বড় child 6-এর সঙ্গে swap	[6, 5, 4]	**11**

h	Insert 7	index 3-এ বসে; parent 5-এর সঙ্গে swap, তারপর root 6-এর সঙ্গে swap	[7, 6, 4, 5]	—
i	Extract-Root	7 বের হয়; 5 root-এ আসে; বড় child 6-এর সঙ্গে swap	[6, 5, 4]	**7**
j	Extract-Root	6 বের হয়; 4 root-এ আসে; child 5 > 4, তাই swap	[5, 4]	**6**



চিত্র ১.১: প্রতিটি অপারেশনের পর Max Heap (লাল শিরোনাম = Extract, তীরের পরে ফেরত মান)

উত্তর: Extract-Root অপারেশনগুলো ক্রমানুসারে **8, 11, 7, 6** ফেরত দেয়। শেষে heap-এ থাকে [5, 4]।

প্রশ্ন ২ 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.

যুক্তি: একটি S-record ও একটি U-record তখনই সম্পর্কিত, যখন একটি সাধারণ T-record (tid) থাকে, যেখানে R-এ (sid, tid) আছে এবং Q-তে (tid, uid) আছে। তাই R ও Q-কে `tid`-এর ওপর join করতে হবে।

```
-- সংক্ষিপ্ত রূপ (R.sid ও Q.uid foreign key, তাই S ও U join না করলেও চলে)
SELECT R.sid, Q.uid
FROM   R
JOIN   Q ON R.tid = Q.tid;

-- পূর্ণ রূপ (S ও U-তে সত্যিই record আছে কিনা তাও নিশ্চিত করে)
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;
```

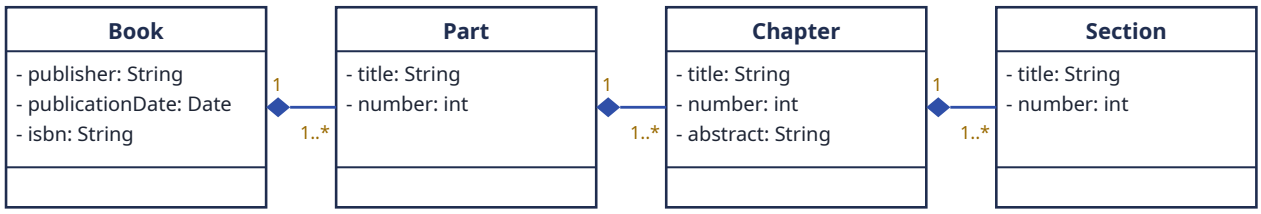
কেন DISTINCT নয়, আর তার ফল কী

একই (sid, uid) জোড়া একাধিক tid-এর মাধ্যমে যুক্ত হলে ফলাফলে একাধিকবার আসবে (যেমন s1–t1–u1 ও s1–t2–u1)। প্রশ্নে DISTINCT নিষেধ, তাই এই পুনরাবৃত্তি গ্রহণযোগ্য। উত্তরে এক লাইনে এটা উল্লেখ করলে পরীক্ষক বুঝবেন আপনি join-এর আচরণ জানেন।

প্রশ্ন ৩ 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" মানে **Composition** (ভরাট diamond ◆)। অংশটি সম্পূর্ণের বাইরে আলাদাভাবে টিকে থাকে না: বই মুছে ফেললে তার part, chapter ও section-ও মুছে যায়। Multiplicity হলো সম্পূর্ণের দিকে **1** এবং অংশের দিকে **1..*** ("several")।



চিত্র ১.২: Book–Part–Chapter–Section class diagram (composition)

ব্যাখ্যা লিখুন: (১) চারটি class এবং প্রতিটির attribute; (২) সম্পর্কগুলো composition, কারণ chapter কেবল একটি part-এর ভেতরেই থাকে; (৩) প্রতিটি সম্পর্কের multiplicity 1 → 1..*। চাইলে operation যোগ করতে পারেন, যেমন `Book.addPart()` বা `Chapter.getSections()`।

প্রশ্ন ৪(ক) 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)

ধাপ ১: FP = 180 × 0.87 = **156.6**
ধাপ ২: ধরে নিচ্ছি "performance factor 4" মানে productivity, অর্থাৎ **৪ FP প্রতি person-day**।
Effort = 156.6 ÷ 4 = **39.15** ≈ **40 person-days**

অনুমান স্পষ্ট লিখুন

প্রশ্নে "performance factor"-এর একক দেওয়া নেই। যদি এর অর্থ "প্রতি FP-এ ৪ person-day" হয়, তাহলে Effort = 156.6 × 4 = 626.4 person-days। পরীক্ষার খাতায় এক লাইনে "Assuming productivity = 4 FP/person-day" লিখে প্রথম উত্তরটি দিন। VAF সাধারণত সূত্র থেকে আসে, VAF = 0.65 + 0.01 × ΣF_i; এখানে তা সরাসরি দেওয়া আছে।

প্রশ্ন ৪(খ) 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.

লজিক: প্রথম উপাদানকে min ধরুন। বাকি উপাদানগুলোর ওপর একবার লুপ চালান; কোনোটি min-এর চেয়ে ছোট হলে min আপডেট করুন। 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
```

C-তে array পাঠালে আসলে pointer যায়, তাই size (n) আলাদা parameter হিসেবে পাঠাতে হয়। Java-তে `arr.length` থেকে size পাওয়া যায়।

প্রশ্ন ৫(ক) 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?

সার্ভারের certificate-এ যে public key আছে, তার জোড়া private key দিয়েই সার্ভার প্রমাণ করে "আমিই আসল সাইট"। এই key চুরি হলে আক্রমণকারী যা করতে পারে:

1. **সার্ভারের ছদ্মবেশ (Impersonation):** আসল ও বৈধ certificate-সহ একটি ভুয়া সাইট চালাতে পারে। DNS spoofing, ARP spoofing বা ভুয়া Wi-Fi-এর মাধ্যমে ব্যবহারকারীকে সেখানে নিলে ব্রাউজার কোনো সতর্কবার্তা দেখাবে না।
2. **Active Man-in-the-Middle:** ক্লায়েন্ট ও আসল সার্ভারের মাঝে বসে উভয় দিকের TLS সংযোগ নিজেই শেষ (terminate) করতে পারে। এতে password, OTP, কার্ড নম্বর পড়তে পারে, এবং লেনদেনের পরিমাণ বা গন্তব্য অ্যাকাউন্ট বদলে দিতে পারে।
3. **রেকর্ড করা ট্রাফিক ডিক্রিপ্ট করা (passive):** সার্ভার **RSA key exchange** ব্যবহার করলে (TLS 1.2-এর non-forward-secret cipher suite) session key ক্লায়েন্টের পাঠানো premaster secret থেকে আসে, যা ওই public key দিয়ে এনক্রিপ্ট করা। তাই আগে রেকর্ড করা এবং ভবিষ্যতের সব সেশন ডিক্রিপ্ট করা যায়।
   (EC)DHE (Forward Secrecy, TLS 1.3-এ বাধ্যতামূলক) ব্যবহার হলে private key কেবল handshake-এ signing-এর কাজে লাগে। তখন অতীতের রেকর্ড করা ট্রাফিক ডিক্রিপ্ট করা যায় না, তবে active MitM (পয়েন্ট ২) তখনও সম্ভব।
4. **সার্ভারের নামে সই করা:** একই key দিয়ে অন্য কিছুতে (যেমন code বা message) সই হয়ে থাকলে আক্রমণকারী সার্ভারের নামে জাল সই তৈরি করতে পারে।

প্রতিরোধ ও প্রতিকার	কেন

Certificate অবিলম্বে revoke করা (CRL/OCSP) ও নতুন key pair তৈরি করা	চুরি হওয়া key দিয়ে আর বিশ্বাসযোগ্যতা পাওয়া যাবে না
শুধু ECDHE/TLS 1.3 চালু রাখা	Forward secrecy নিশ্চিত হয়
Private key HSM-এ রাখা	Key HSM থেকে বের করা যায় না
Certificate Transparency monitoring	ভুয়া certificate ধরা পড়ে
কম মেয়াদের certificate, নিয়মিত key rotation	চুরি হলেও ক্ষতির সময়সীমা কম থাকে

প্রশ্ন ৫(খ) 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?

Generator-এ ১১টি বিট আছে, তাই এর degree **r = 11 − 1 = 10** (CRC remainder ১০ বিটের)।

Burst length L	Detection probability
$L \le r$	১০০% (সব ধরা পড়ে)
$L = r + 1$	$1 - (1/2)^{r-1}$
$L > r + 1$	$1 - (1/2)^{r}$

এখানে L = 10 এবং r = 10, তাই $L \le r$। Generator-এর সর্বোচ্চ ও সর্বনিম্ন (x^0) বিট দুটিই 1, তাই **সব burst error ধরা পড়বে; detection probability = 1 (100%)**।

তুলনার জন্য: L = 11 হলে probability = $1 - 1/2^9 \approx 0.998$; $L \ge 12$ হলে $1 - 1/2^{10} \approx 0.999$।

প্রশ্ন ৬ 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.

বিষয়	Adjacency List	Adjacency Matrix
Space	O(V + E): sparse graph-এ ভালো	$O(V^2)$
u-এর সব neighbour বের করা	O(out-degree(u))	O(V), পুরো row স্ক্যান করতে হয়
(u, v) edge আছে কিনা	O(out-degree(u))	**O(1)**
BFS / DFS	**O(V + E)**	$O(V^2)$
In-degree বের করা	O(V + E)	O(V), একটি column

List-এ দ্রুত: ১০ লক্ষ নোড ও ৫০ লক্ষ edge-এর একটি sparse graph-এ (যেমন ব্যাংক গ্রাহকের লেনদেন নেটওয়ার্ক) BFS/DFS দিয়ে সব reachable নোড বের করা, বা প্রতিটি নোডের out-degree গোনা। List-এ এতে O(V + E) সময় লাগে, matrix-এ $O(V^2)$।

Matrix-এ দ্রুত: "u থেকে v-তে সরাসরি edge আছে কিনা" বারবার জিজ্ঞাসা করা (O(1) lookup), dense graph-এ Floyd–Warshall all-pairs shortest path, অথবা কোনো vertex **universal sink** কিনা (in-degree V−1, out-degree 0) তা O(V) সময়ে বের করা।

প্রশ্ন ৭(ক) 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?

এটি ব্রাউজারের **frame navigation policy** নিয়ে প্রশ্ন। বর্তমান ব্রাউজার **Descendant Policy** মেনে চলে: একটি frame কেবল তার নিজের ভেতরের (descendant) frame-কে অন্য URL-এ নিতে পারে।

1. **Pixel delegation-এর যুক্তি:** B যদি A-এর display area-র ভেতরে থাকে, তাহলে স্ক্রিনের ওই অংশটি A-রই দেওয়া। A চাইলে এমনিতেই ওই জায়গায় নিজের কনটেন্ট দেখাতে, B-কে সরিয়ে দিতে বা ওপরে কিছু বসাতে পারে। তাই B-কে অন্য origin-এ পাঠানোর অনুমতি দিলে A-এর **নতুন কোনো ক্ষমতা যোগ হয় না**, আর নিরাপত্তাও কমে না।
2. **Same-Origin Policy অক্ষত থাকে:** Navigation করলে A, B-এর DOM, cookie বা ডেটা **পড়তে পারে না**; শুধু B-এর জায়গায় কী লোড হবে তা ঠিক করে। তাই confidentiality ভাঙে না।
3. **উদার নীতির ঝুঁকি:** যেকোনো frame যেকোনো frame-কে navigate করতে পারলে (permissive policy) একটি আক্রমণকারী frame অন্য সাইটের ভেতরের login বা payment frame-কে ভুয়া পেজে পাঠিয়ে দিতে পারত (frame hijacking / Guninski attack)। ব্যবহারকারী ঠিকানা দেখে আসল সাইট ভাবত, অথচ তথ্য যেত আক্রমণকারীর কাছে।
4. **তাই সীমারেখা:** "যার স্ক্রিন, তার নিয়ন্ত্রণ।" A কেবল নিজের এলাকার ভেতরের frame বদলাতে পারে। পাশের বা ওপরের অন্য origin-এর frame বদলাতে পারে না।

প্রশ্ন ৭(খ) 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.

LRU (Least Recently Used) নিয়মে page fault হলে, সবচেয়ে বেশি সময় ধরে ব্যবহার হয়নি এমন page সরানো হয়। 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

সরানোর সিদ্ধান্ত:

- Ref 4-এ সবচেয়ে পুরোনো 1, তাই 1 সরে।
- Ref 1 (৮ম)-এ ক্রম 2, 3, 4 (২ সবচেয়ে পুরোনো), তাই 2 সরে।
- Ref 2 (৯ম)-এ ক্রম 3, 4, 1, তাই 3 সরে।
- Ref 3 (১২তম)-এ ক্রম 4, 2, 1, তাই 4 সরে।
- Ref 4 (১৪তম)-এ ক্রম 2, 3, 1, তাই 2 সরে।

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

১.২ Bangladesh Bank — Assistant Programmer (০৩.০২.২০২৩)

তথ্য	বিবরণ

তারিখ	০৩.০২.২০২৩ (১ ঘণ্টা MCQ + ২ ঘণ্টা লিখিত)
প্রশ্নকর্তা	BIBM
ধরন	৮০ MCQ; লিখিত = ৯টি subjective (১৫০) + গণিত ৪টি (২০) + Focus (১০) + অনুবাদ B/E (২০)
টপিক	Hashing, graph representation, LCM program, SQL, UML, overriding/hiding, CIA ও E2E encryption, Diffie-Hellman MitM

প্রশ্ন ১ 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 খালি	**10**
3	3	3 খালি	**3**
5	5	5 খালি	**5**
6	6	6 খালি	**6**
16	3	3 (3 দিয়ে ভরা) → 4 খালি	**4**
17	4	4 (16) → 5 (5) → 6 (6) → 7 খালি	**7**
19	6	6 (6) → 7 (17) → 8 খালি	**8**

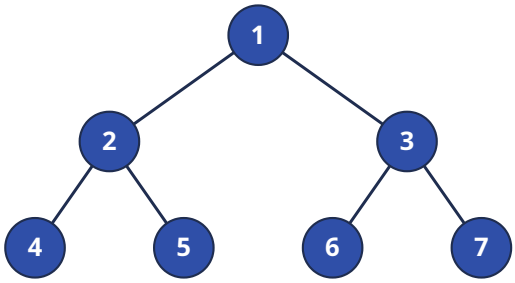
Index	0	1	2	3	4	5	6	7	8	9	10	11	12
Key	–	–	–	3	16	5	6	17	19	–	10	–	–

মোট collision: 16-এ ১টি, 17-এ ৩টি, 19-এ ২টি, মোট ৬টি। Index 3 থেকে 8 পর্যন্ত একটানা ভরা অংশটি **primary clustering**-এর উদাহরণ, যা linear probing-এর প্রধান দুর্বলতা। এর প্রতিকার হলো quadratic probing বা double hashing। Load factor $\alpha = 7/13 \approx 0.54$।

প্রশ্ন ২ 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.

Binary heap-এর মতো নম্বর দিলে vertex i-এর children হয় **2i** ও **2i+1**, আর parent হয় $\lfloor \mathbf{i/2} \rfloor$। Tree-কে undirected graph ধরা হলো।



চিত্র ১.৩: ৭ শীর্ষের complete binary tree (heap numbering)

Adjacency List

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

Directed (parent → child) হিসেবে ধরলে: 1→2,3; 2→4,5; 3→6,7; এবং 4–7-এর list খালি।

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

Edge সংখ্যা = V − 1 = 6। List-এ মোট entry ২ × 6 = 12, matrix-এ cell 7 × 7 = 49, যার মাত্র 12টি 1। Tree sparse, তাই এখানে list বেশি সাশ্রয়ী।

প্রশ্ন ৩ 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

Overflow এড়াতে `(a / gcd) * b` ক্রমে হিসাব করুন। এতে আগে ভাগ হয়, তারপর গুণ।

```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))।

প্রশ্ন 8 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.

যুক্তি: "Romantic comedy" মানে একই movie-র দুটি genre আছে: **Romance** এবং **Comedy**। প্রতিটি genre HasGenre-এ আলাদা সারিতে থাকে, তাই দুটি শর্ত একই সারিতে মেলানো যায় না।

```sql
-- পদ্ধতি ১: GROUP BY + HAVING (সবচেয়ে পরিষ্কার)
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;

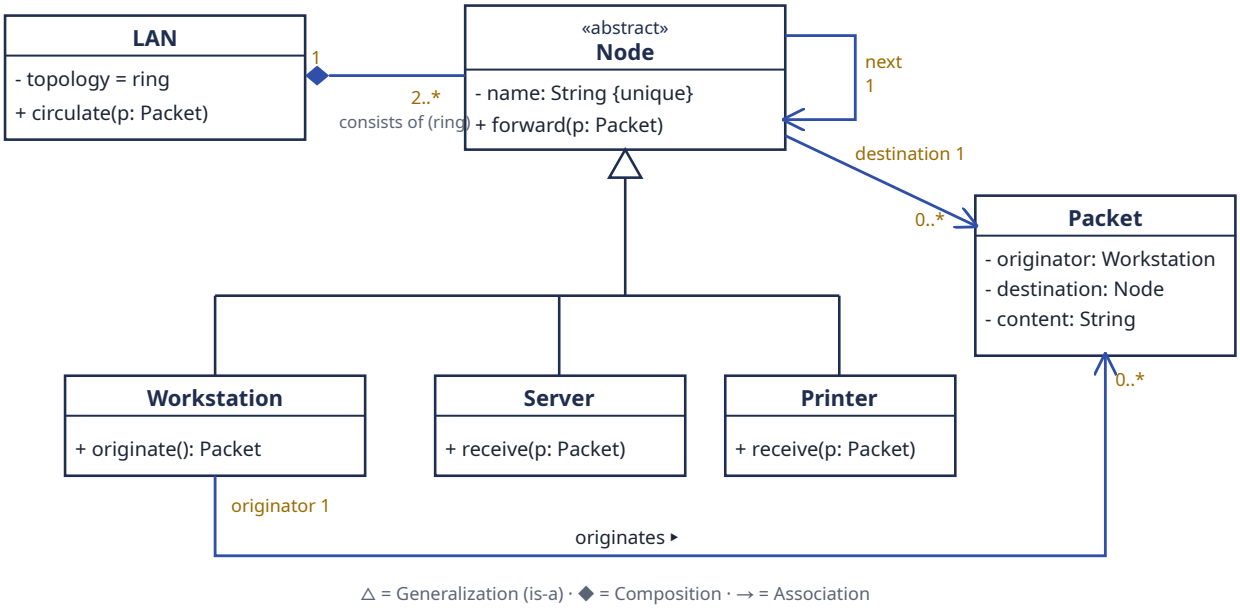
-- পদ্ধতি ২: দুটি IN subquery
SELECT COUNT(*) AS romantic_comedy_count
FROM   Movies m
WHERE  m.mid IN (SELECT hg.mid FROM HasGenre hg JOIN Genres g ON g.gid = hg.gid
                 WHERE g.genre = 'Romance')
  AND  m.mid IN (SELECT hg.mid FROM HasGenre hg JOIN Genres g ON g.gid = hg.gid
                 WHERE g.genre = 'Comedy');
```

সাধারণ ভুল

`WHERE genre = 'Romance' AND genre = 'Comedy'` লিখলে সবসময় **0** আসে, কারণ এক সারির genre একসাথে দুটি মান হতে পারে না। আবার `OR` লিখলে শুধু romance বা শুধু comedy movie-ও গোনা হয়। যদি ডেটাবেসে "Romantic Comedy" নামে একটিমাত্র genre থাকে, তাহলে সহজ `WHERE g.genre = 'Romantic Comedy'` দিয়েই হবে। উত্তরে এই অনুমানটি উল্লেখ করুন।

প্রশ্ন ৫ 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.



চিত্র ১.৪: Token-ring LAN-এর class diagram

বাক্য	মডেলে কী
"LAN consists of nodes; circular configuration"	LAN ◆— Node, composition, multiplicity 1 → 2..*; ring গঠন হয় self-association দিয়ে
"Every node has a unique name and refers to its next node"	Node-এ `name {unique}`; self-association `next`, multiplicity 1
"Different kinds of nodes"	Node একটি abstract superclass; Workstation, Server ও Printer তার subclass (generalization)
"Workstations are originators of messages"	Workstation 1 —originates→ 0..* Packet
"Servers and printers can receive messages"	Server ও Printer-এ `receive(p: Packet)` operation
"Packets contain originator, destination and content"	Packet-এর attribute বা association: originator (Workstation), destination (Node), content

প্রশ্ন ৬ 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) {}
}
```

ধারণা	শর্ত	কখন resolve হয়
Overloading	একই নাম, ভিন্ন parameter list (একই বা subclass-এ)	Compile time
Overriding	Subclass-এ একই signature-এর **instance** method; parent-এর method-ও instance	Runtime (dynamic dispatch)
Hiding	Parent ও child **দুজনেরই static** method, একই signature	Compile time (reference type দেখে)

Method	A-তে	B-তে	ফল
m1()	instance	static	**Compile error**: "m1() in B cannot override m1() in A; overriding method is static"। Static method দিয়ে instance method লুকানো যায় না।

m2()	instance	instance, একই signature	**Overridden**
m3(int)	instance	instance, একই signature	**Overridden**
m4(String)	instance	static	**Compile error** (m1-এর একই কারণে)

উপসংহার: m2 ও m3 **overridden**। m1 ও m4 compile হবে না, তাই এখানে কোনো **hiding** ঘটে না। কোনো method-এর parameter list ভিন্ন নয়, তাই কোনো **overloading**-ও নেই। এছাড়া কোডে দুটি সমস্যা ঠিক করতে হবে: `extends a` হবে `extends A`, আর একটি .java ফাইলে দুটি public class থাকতে পারে না, তাই দুটি আলাদা ফাইলে রাখতে হবে।

Hiding ও overloading কখন হতো: A-এর m1 যদি `public static void m1()` হতো, তাহলে B-এর static m1() সেটিকে **hide** করত। B-তে যদি `public void m3(double d)` থাকত, তাহলে সেটি A-এর m3(int)-এর **overload** হতো।

```
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  : runtime-এ object-এর type (B) দেখে
        ref.s();   // A.s  : compile time-এ reference-এর type (A) দেখে, অর্থাৎ hiding
    }
}
```

প্রশ্ন ৭ 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:** অননুমোদিত কেউ তথ্য পড়তে পারবে না।
- **Integrity:** তথ্য অননুমোদিতভাবে বদলানো যাবে না।
- **Availability:** অনুমোদিত ব্যবহারকারী প্রয়োজনের সময় তথ্য পাবে।

End-to-end encryption (E2EE) মূলত **confidentiality** ও (MAC বা AEAD ব্যবহার করলে) **integrity** নিশ্চিত করে।

উত্তর: Application Layer (Layer 7)।

লেয়ার	উদাহরণ	E2E কি?	খরচ ও রক্ষণাবেক্ষণ
Network (L3)	IPsec	না; host-to-host বা gateway-to-gateway সুরক্ষা দেয়, ব্যবহারকারী-থেকে-ব্যবহারকারী নয়	OS ও router কনফিগার করতে হয়; বিশেষজ্ঞ লাগে
Transport (L4)	TLS	না; প্রতিটি hop-এ (সার্ভার, proxy, load balancer) ডিক্রিপ্ট হয়, সার্ভার plaintext দেখে	সহজ, কিন্তু উদ্দেশ্য পূরণ হয় না

Application (L7)	Signal Protocol, PGP, libsodium	**হ্যাঁ**; কেবল প্রেরক ও প্রাপকের অ্যাপের কাছে key থাকে	**প্রস্তুত লাইব্রেরি, নেটওয়ার্ক অবকাঠামো বদলাতে হয় না**

কারণ:

1. **Development time:** পরীক্ষিত লাইব্রেরি (OpenSSL, libsodium, Bouncy Castle) সরাসরি অ্যাপের কোডে ব্যবহার করা যায়। Kernel বা নেটওয়ার্ক ডিভাইসে হাত দিতে হয় না।
2. **Maintainability:** এনক্রিপশন লজিক অ্যাপের ভেতরেই থাকে। নেটওয়ার্ক, ক্লাউড প্রোভাইডার বা ISP বদলালেও কিছু পরিবর্তন করতে হয় না, এবং আপডেট সাধারণ সফটওয়্যার রিলিজের মতো দেওয়া যায়।
3. **Development cost:** বিশেষ হার্ডওয়্যার (VPN concentrator) বা প্রতিটি router কনফিগারেশনের খরচ লাগে না।
4. **প্রকৃত end-to-end:** একমাত্র এই লেয়ারেই মাঝের সার্ভার, এমনকি কোম্পানির নিজের সার্ভারও, ডেটা পড়তে পারে না। ফলে ডেটা লিক হলেও confidentiality বজায় থাকে।

সীমাবদ্ধতা: প্রতিটি অ্যাপে আলাদাভাবে বাস্তবায়ন করতে হয় এবং key management (key exchange, backup) সাবধানে ডিজাইন করতে হয়। বাস্তবে Application-layer E2EE-র নিচে TLS-ও রাখা হয় (defense in depth)।

> **প্রশ্ন ৮** 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.*

মূল দুর্বলতা: সাধারণ Diffie-Hellman-এ **authentication নেই**। কেউ যাচাই করে না যে প্রাপ্ত public key আসলে অপর পক্ষেরই। প্রকাশ্য মান: prime q ও primitive root α।

আক্রমণের ধাপ (আক্রমণকারী Darth, Stallings-এর বিবরণ অনুযায়ী):

1. Darth দুটি private key X_{D1} ও X_{D2} বেছে নেয় এবং সংশ্লিষ্ট public key হিসাব করে: $Y_{D1} = \alpha^{X_{D1}} \bmod q$, $Y_{D2} = \alpha^{X_{D2}} \bmod q$।
2. Alice, Bob-কে Y_A পাঠায়। Darth সেটি আটকে দিয়ে Bob-কে Y_{D1} পাঠায়, এবং নিজে হিসাব করে $K2 = Y_A^{X_{D2}} \bmod q$।
3. Bob, Y_{D1} পেয়ে হিসাব করে $K1 = Y_{D1}^{X_B} \bmod q$।
4. Bob, Alice-কে Y_B পাঠায়। Darth সেটি আটকে দিয়ে Alice-কে Y_{D2} পাঠায়, এবং হিসাব করে $K1 = Y_B^{X_{D1}} \bmod q$।
5. Alice, Y_{D2} পেয়ে হিসাব করে $K2 = Y_{D2}^{X_A} \bmod q$।

ফল: Alice ও Darth একই key K2 পায়, আর Bob ও Darth একই key K1 পায়। Alice ও Bob ভাবে তারা নিজেদের মধ্যে একটি গোপন key ভাগ করেছে। Darth প্রতিটি বার্তা ডিক্রিপ্ট করে পড়তে বা বদলাতে পারে, তারপর আবার এনক্রিপ্ট করে সামনে পাঠায়।

সংখ্যাগত উদাহরণ: q = 23, α = 5

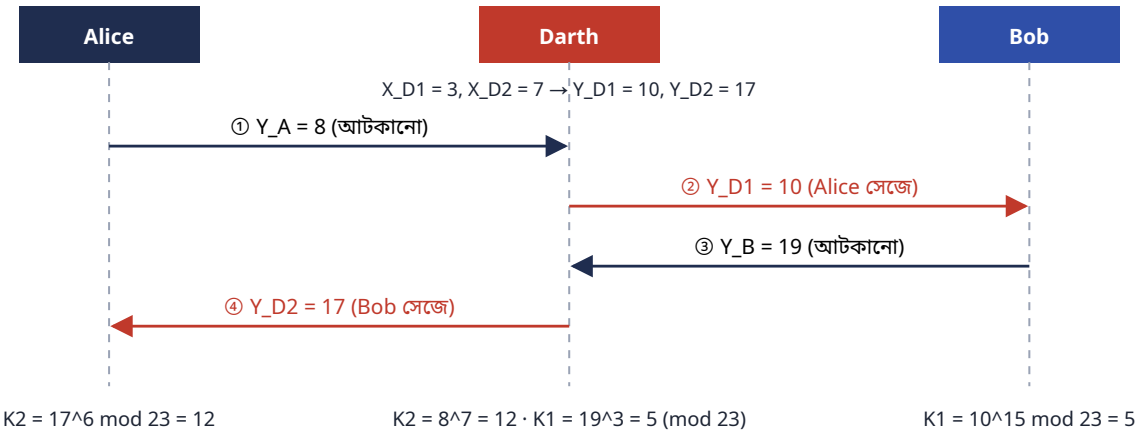
Alice: $X_A = 6 \rightarrow Y_A = 5^6 \bmod 23 =$ **8** | Bob: $X_B = 15 \rightarrow Y_B = 5^{15} \bmod 23 =$ **19**

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

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

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

আক্রমণ না থাকলে Alice ও Bob একই key পেত: $19^6 \bmod 23 = 8^{15} \bmod 23 = 2$।



চিত্র ১.৫: দুটি key pair ব্যবহার করে Diffie-Hellman-এর ওপর MitM আক্রমণ

প্রতিরোধ:

- Public key-কে **authenticate** করা: digital signature বা certificate-সহ DH (Station-to-Station protocol, TLS-এর signed ECDHE)।
- বাইরের কোনো চ্যানেলে key fingerprint মিলিয়ে দেখা (যেমন Signal-এর safety number)।
- Pre-shared key বা PKI ব্যবহার করা।

১.৩ Combined 5 Bank (Job ID 10198) — Assistant Programmer (০৯.০২.২০২৪)

তথ্য	বিবরণ
ব্যাংক	Sonali Bank, BKB, Karmasangsthan Bank, ICB প্রমুখ
তারিখ	০৯.০২.২০২৪ (১ ঘণ্টা MCQ + ২ ঘণ্টা লিখিত)
প্রশ্নকর্তা	BIBM
ধরন	৮০ MCQ (IT ৫০ + গণিত ও GK ২০); লিখিত = ১০টি subjective (১৫০) + গণিত ৪টি (২০) + Focus (২০) + Letter (১০)
বৈশিষ্ট্য	**প্রোগ্রামিং-প্রধান**: ৩টি প্রোগ্রাম লেখা, ১টি output নির্ণয়, ৩টি SQL, এরপর তত্ত্ব ও নিজের প্রজেক্ট নিয়ে প্রশ্ন

> **প্রশ্ন ১** 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.*

লজিক: x prime হবে যদি x ≥ 2 হয় এবং 2 থেকে √x পর্যন্ত কোনো সংখ্যা দিয়ে x বিভাজ্য না হয়। লুপের শর্ত `d * d <= x` লিখলে sqrt function লাগে না। প্রতিটি সংখ্যার জন্য O(√x), মোট O(n√n)। বড় n-এর জন্য Sieve of Eratosthenes দ্রুততর: 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 prime নয়, তাই লুপ 2 থেকে শুরু।

প্রশ্ন ২ 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: পাশাপাশি পদের পার্থক্য 1, 2, 3, 4, ...। তাই প্রতিটি পদ আগের পদের সঙ্গে k যোগ করে পাওয়া যায় (k = 1, 2, 3, ...)।

k-তম পদ: $t_k = 1 + k(k-1)/2$ | n পদের যোগফল: $S_n = n + (n-1)n(n+1)/6$

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

```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`। n = 7 হলে পদগুলো 16 ও 22 পর্যন্ত যায়, Sum = 63 (সূত্রে 7 + 6·7·8/6 = 63 ✔)।

প্রশ্ন ৩ 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 বিশ্লেষণ: i-তম সারিতে i টি অঙ্ক। বিজোড় সারি 1 দিয়ে আর জোড় সারি 0 দিয়ে শুরু হয়, তারপর 1 ও 0 পালাক্রমে আসে। দুটি নিয়মকে একসাথে লেখা যায়: **(i + j) জোড় হলে 1, বিজোড় হলে 0** (i ও j 1 থেকে গোনা)।

```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);
        }
    }
}
```

যাচাই: i = 1: j = 1 → 2 জোড় → "1"। i = 2: 3 → 0, 4 → 1 → "01"। i = 4: 5, 6, 7, 8 → "0101"। প্রশ্নের নমুনার সঙ্গে হুবহু মেলে ✔। প্রচলিত সংখ্যাভিত্তিক Floyd's triangle (1 / 2 3 / 4 5 6 ...) চাইলে একটি counter রেখে প্রতিবার `printf("%d ", c++)` লিখুন।

প্রশ্ন 8 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);
}
```

মনে রাখুন: `i--` (post-decrement) আগে মান ব্যবহার করে, তারপর কমায়। `--i` (pre-decrement) আগে কমায়, তারপর মান ব্যবহার করে। তাই প্রতিটি call-এ i মোট ২ কমে। দ্বিতীয় printf recursive call ফেরার পরে চলে, তাই ওই মানগুলো **উল্টো ক্রমে** ছাপা হয়।

Call	প্রথম printf	i-এর পরিবর্তন	Recursive call	ফেরার পর দ্বিতীয় printf
fun(5)	"5, "	i-- → 4, --i → 3	fun(3)	"3, " (সবশেষে)
fun(3)	"3, "	i-- → 2, --i → 1	fun(1)	"1, "
fun(1)	"1, "	i-- → 0, --i → −1	fun(−1)	"−1, "
fun(−1)	—	i ≤ 0, তাই কিছু না ছেপেই return		

ক্রম: নামার পথে 5, 3, 1 → fun(−1) ফেরে → ওঠার পথে −1 (fun(1)), 1 (fun(3)), 3 (fun(5))।

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

সূক্ষ্ম বিষয় (উত্তরে এক লাইনে লিখুন)

`int fun()` একটি non-void function, অথচ `return;` -এ কোনো মান নেই এবং শেষেও return নেই। C99/C11-এ এটি constraint violation বা warning, কিন্তু যেহেতু ফেরত মানটি কোথাও ব্যবহার হয়নি, বেশিরভাগ compiler (gcc) warning দিয়ে উপরের output-ই দেয়। প্রশ্নপত্রের "I", "Printf" ও "Ø" OCR বা টাইপের ভুল; সেগুলো `i`, `printf` ও `0` ধরা হয়েছে।

প্রশ্ন ৫–৬ 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 সংশোধন (উত্তরে উল্লেখ করুন): Employee-র DeptCode-এর type Department-এর DeptCode-এর মতোই `INT` হওয়া উচিত, যাতে join ও foreign key ঠিকমতো কাজ করে। মূল DDL-এ `Commission INT` -এর পরে কমা নেই এবং শেষে একটি বাড়তি কমা আছে; দুটোই syntax error।

```
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; Commission NULL হলে 0 ধরা হয়
SELECT EmpFname,
       Salary + COALESCE(Commission, 0) AS TotalSalary
FROM   Employee;

-- (b) গড় বেতনের চেয়ে বেশি পান যাঁরা, তাঁদের নাম ও Location
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) প্রতিটি EmpFname-এ কতজন কর্মচারী, সংখ্যার ক্রমে
SELECT   EmpFname, COUNT(EmpCode) AS EmpCount
FROM     Employee
GROUP BY EmpFname
ORDER BY EmpCount ASC;
```

ব্যাখ্যা ও বিকল্প

(a): `NULL + সংখ্যা = NULL`, তাই COALESCE (বা MySQL-এ IFNULL, SQL Server-এ ISNULL) না দিলে কমিশনহীন কর্মচারীর total salary NULL দেখাবে।

(b): WHERE-এ সরাসরি `AVG()` ব্যবহার করা যায় না, তাই subquery লাগে।

(c): প্রশ্নটি অস্পষ্ট। Aggregate-এর সঙ্গে non-aggregate কলাম (EmpFname) রাখতে হলে সেটি GROUP BY-তে থাকতে হবে। উপরে "একই নামে কতজন" ধরা হয়েছে। যদি উদ্দেশ্য হয় প্রতিটি বিভাগে কতজন কর্মচারী, তাহলে:

```
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;
```

আর প্রতিটি নামের পাশে মোট কর্মচারী সংখ্যা দেখাতে চাইলে window function: `SELECT EmpFname, COUNT(*) OVER () AS TotalEmp FROM Employee ORDER BY EmpCode;`। খাতায় নিজের ব্যাখ্যাটি এক লাইনে লিখে দিন।

প্রশ্ন ৭ Combined 5 Bank AP 2024

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

বিষয়	Stack	Queue

নীতি	**LIFO** (Last In First Out)	**FIFO** (First In First Out)
প্রান্ত	একটি প্রান্ত (top)। এখানেই insert ও delete হয়	দুটি প্রান্ত: rear-এ insert, front-এ delete
অপারেশন	push, pop, peek	enqueue, dequeue, front
Pointer	top	front ও rear
Complexity	push/pop O(1)	enqueue/dequeue O(1) (circular array বা linked list-এ)
রূপভেদ	—	Circular queue, priority queue, deque
বাস্তব উদাহরণ	প্লেটের স্তূপ	ব্যাংকের কাউন্টারের লাইন

Stack-এর ব্যবহার

1. Function call ও recursion-এর call stack, যেখানে return address ও local variable থাকে।
2. Expression evaluation ও infix → postfix রূপান্তর; balanced parentheses যাচাই।
3. (অতিরিক্ত) Undo/Redo, ব্রাউজারের Back বাটন, DFS।

Queue-এর ব্যবহার

1. CPU scheduling (Round Robin ready queue) ও printer spooling।
2. BFS traversal; নেটওয়ার্ক router-এর packet buffer।
3. (অতিরিক্ত) ব্যাংকের টোকেন সিস্টেম, message queue (RabbitMQ, Kafka)।

> **প্রশ্ন ৮** Combined 5 Bank AP 2024
>
> *What is polymorphism? Describe the different types of polymorphism.*

সংজ্ঞা: Polymorphism (poly = অনেক, morph = রূপ) মানে একই interface বা নাম দিয়ে ভিন্ন ভিন্ন type-এর object-এর সঙ্গে ভিন্ন আচরণ করা। এটি OOP-এর চারটি স্তম্ভের একটি; বাকি তিনটি encapsulation, inheritance ও abstraction।

ধরন	অন্য নাম	কীভাবে	Binding
Compile-time	Static polymorphism	Method overloading (একই নাম, ভিন্ন parameter); C++-এ operator overloading	Early/static binding
Runtime	Dynamic polymorphism	Method overriding + upcasting (parent reference, child object); C++-এ virtual function	Late/dynamic binding
Parametric	Generics / Templates	Java-র `List<T>`, C++ template। একই কোড বিভিন্ন type-এর সঙ্গে কাজ করে	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)                    // একই call, ভিন্ন আচরণ
            System.out.println(a.interest(10000)); // 500.0, 900.0
    }
}
```

সুবিধা: নতুন account type যোগ করলে Bank-এর লুপ বদলাতে হয় না (Open–Closed principle)। কোড পুনর্ব্যবহারযোগ্য ও extensible হয়।

> **প্রশ্ন ৯** Combined 5 Bank AP 2024
>
> *Describe the different types of relationships in a DBMS.*

ER model-এ **relationship** মানে দুই বা ততোধিক entity-র মধ্যে সম্পর্ক। একে তিনভাবে শ্রেণিবিভাগ করা হয়।

(ক) Cardinality (mapping) অনুযায়ী

ধরন	অর্থ	উদাহরণ	টেবিলে বাস্তবায়ন
One-to-One (1:1)	A-এর একটি record, B-এর সর্বোচ্চ একটির সঙ্গে	Customer — NID; Employee — Desk	যেকোনো এক পাশে UNIQUE foreign key
One-to-Many (1:N)	A-এর একটি, B-এর অনেকের সঙ্গে	Branch — Accounts; Department — Employees	"Many" পাশের টেবিলে foreign key
Many-to-One (N:1)	1:N-এর উল্টো দিক থেকে দেখা	Accounts — Branch	একই
Many-to-Many (M:N)	দুই পাশেই অনেক	Customer — Account (joint account); Student — Course	**Junction (bridge) টেবিল**, দুই FK মিলিয়ে composite PK, যেমন প্রশ্ন ২-এর R ও Q

(খ) Degree অনুযায়ী

- **Unary (recursive):** একই entity নিজের সঙ্গে সম্পর্কিত, যেমন Employee *manages* Employee, বা Node *next* Node (token ring)।
- **Binary:** দুটি entity, যেমন Customer *holds* Account। এটিই সবচেয়ে প্রচলিত।
- **Ternary:** তিনটি entity একসাথে, যেমন Doctor–Patient–Medicine (*prescribes*), বা Supplier–Part–Project।

(গ) Participation অনুযায়ী

- **Total (mandatory):** প্রতিটি record-কে সম্পর্কে থাকতেই হবে, যেমন প্রতিটি Loan-এর একজন Borrower থাকবেই। ER diagram-এ এটি দুই দাগ দিয়ে দেখানো হয়।
- **Partial (optional):** সব record-এর থাকা জরুরি নয়, যেমন সব Customer-এর Loan নেই।

"Relation" শব্দটি দিয়ে যদি টেবিল বোঝানো হয়: base relation (সংরক্ষিত টেবিল), view (virtual relation) ও query result (derived relation)। প্রয়োজনে উত্তরের শেষে এক লাইনে যোগ করুন।

প্রশ্ন ১০ 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?

কৌশল

নিজের থিসিস, ইন্টার্নশিপ বা চাকরির প্রজেক্ট নিন এবং নিচের কাঠামোয় লিখুন। ব্যাংকিং সংশ্লিষ্ট হলে আরও ভালো। পরীক্ষক যাচাই করেন আপনি সমস্যা চিহ্নিত করে **প্রযুক্তিগত সমাধান** দিতে পারেন কিনা। শুধু "অনেক কষ্ট হয়েছিল" লিখলে নম্বর আসে না। প্রতিটি চ্যালেঞ্জের সঙ্গে নির্দিষ্ট টুল বা কৌশলের নাম লিখুন।

মডেল উত্তর (নিজের প্রজেক্ট অনুযায়ী বদলে নিন):

১. প্রজেক্ট পরিচিতি: "Micro-Loan Management System" — একটি ওয়েব অ্যাপ্লিকেশন, যা একটি NGO বা ক্ষুদ্রঋণ প্রতিষ্ঠানের ঋণ আবেদন, অনুমোদন, কিস্তি সংগ্রহ ও রিপোর্ট পরিচালনা করে। ব্যবহারকারী প্রায় ৫০ জন মাঠকর্মী ও ১০ জন শাখা ব্যবস্থাপক; গ্রাহকের রেকর্ড প্রায় ২০,০০০।

২. প্রযুক্তি ও আমার ভূমিকা: Java (Spring Boot) REST API, MySQL, React frontend, Git ও Docker। আমি backend ডেভেলপার হিসেবে database design, কিস্তি হিসাবের মডিউল ও রিপোর্টিং তৈরি করেছি। কাজ চলেছে Agile (২ সপ্তাহের sprint) পদ্ধতিতে।

চ্যালেঞ্জ	কারণ	সমাধান
একই কিস্তি দুবার পোস্ট হওয়া (double posting)	দুর্বল নেটওয়ার্কে মাঠকর্মী বারবার Submit চাপতেন; concurrent request	Transaction (`@Transactional`) ও row-level lock ব্যবহার; প্রতিটি পেমেন্টে idempotency key; ক্লিকের পর বাটন নিষ্ক্রিয়
মাসিক রিপোর্ট খুব ধীর (প্রায় ৪০ সেকেন্ড)	বড় টেবিলে full table scan; N+1 query	`EXPLAIN` দিয়ে বিশ্লেষণ; (branch_id, due_date)-এ composite index; JOIN FETCH; রাতের batch-এ summary টেবিল। ফলে সময় নেমে আসে ২ সেকেন্ডের নিচে
Requirement বারবার বদলানো	সুদের হার ও জরিমানার নিয়ম শাখাভেদে ভিন্ন	নিয়মগুলো কোডে hard-code না করে configuration টেবিলে রাখা; প্রতিটি sprint শেষে demo দিয়ে feedback নেওয়া
নিরাপত্তা	সংবেদনশীল আর্থিক ও ব্যক্তিগত তথ্য	bcrypt দিয়ে password hashing; role-based access (RBAC); prepared statement দিয়ে SQL injection প্রতিরোধ; HTTPS; audit log

Deployment-এ "আমার পিসিতে চলে, সার্ভারে চলে না"	পরিবেশের পার্থক্য	Docker image, CI pipeline (GitHub Actions) এবং স্বয়ংক্রিয় unit ও integration test

৩. ফলাফল ও শিক্ষা: কাগজের খাতার বদলে রিয়েল-টাইম ড্যাশবোর্ড চালু হয় এবং হিসাবের ভুল প্রায় শূন্যে নেমে আসে। শিক্ষা: শুরুতেই ডেটার অখণ্ডতা (transaction, constraint) নিশ্চিত করা, performance মেপে তারপর অপ্টিমাইজ করা, আর ব্যবহারকারীর সঙ্গে নিয়মিত যোগাযোগ রাখা জরুরি। ব্যাংকের Assistant Programmer হিসেবে CBS-সংলগ্ন অ্যাপ বা ইন্টারনাল টুল তৈরিতে এই অভিজ্ঞতা সরাসরি কাজে লাগবে।

অধ্যায় ২ — সমাধানকৃত বিগত প্রশ্ন: ২০২৬ (ও ২০২০)

Solved AP Papers — BSCS 03.01.2026, BSCS 17.01.2026, BSCS 2020

এই অধ্যায়ে সাম্প্রতিক দুটি AP লিখিত পরীক্ষা এবং ২০২০ সালের BSCS Officer (IT) লিখিত পরীক্ষার **শুধু বিষয়ভিত্তিক (technical) প্রশ্নগুলোর** পূর্ণ ১০-নম্বরের মডেল উত্তর দেওয়া হলো। গণিত, ইংরেজি, অনুবাদ, রচনা ও GK অংশ অধ্যায় ৭-এ আছে। প্রশ্নগুলো পরীক্ষার্থীদের স্মৃতি থেকে সংকলিত, তাই OCR/টাইপের ভুল নিঃশব্দে ঠিক করা হয়েছে; কোথাও অনুমান করতে হলে তা স্পষ্ট লেখা আছে।

> **কেন এই তিনটি পেপার গুরুত্বপূর্ণ**
>
> ৩১ অক্টোবরের AP (Job ID-25106) বিজ্ঞপ্তিতে লিখিত অংশ: **Subjects Related ১৫০ + Mathematics ২০ + English Language Proficiency ৩০**। জানুয়ারি ২০২৬-এর AP পেপারে (Paper E) ঠিক এই ধাঁচেই প্রশ্ন ছিল — ১৫টি বিষয়ভিত্তিক, ৪টি গণিত এবং "English Proficiency" নামে আলাদা অংশ (grammar, paragraph, sentence making)। তাই Paper E-কে সবচেয়ে কাছের মডেল ধরে প্রস্তুতি নিন।

২.১ Paper D — BSCS Assistant Programmer / Officer (IT), ০৩.০১.২০২৬

বিষয়	তথ্য
পরীক্ষা	BSCS (Sonali ও অন্যান্য Bank PLC) — Assistant Programmer / Officer (IT)
তারিখ ও সময়	০৩/০১/২০২৬, সকাল ৯টা–১২টা
লিখিত	২০০ নম্বর: Non-tech (বাংলা রচনা, ইংরেজি রচনা, অনুবাদ B→E ও E→B, GK) + Subject Related প্রশ্ন ৬–২০ (১৫ × ১০ = ১৫০)
লক্ষণীয়	সংকলনে প্রশ্ন ১৩/১৫ (new employee account) ও ১৪/১৬ (alarm truth table) দুবার ছাপা হয়েছে — এখানে প্রতিটি একবার, পেপারের ক্রম (৬–২০) অনুযায়ী সাজানো

প্রশ্ন ৬ BSCS AP 03.01.2026

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

OSI (Open Systems Interconnection) Model হলো ISO প্রণীত ৭-স্তরের একটি reference model, যা দুটি কম্পিউটারের মধ্যে ডেটা আদান-প্রদানের কাজকে ভাগ করে প্রতিটি স্তরকে নির্দিষ্ট দায়িত্ব দেয়। প্রেরকের দিকে ডেটা উপর থেকে নিচে নামে (প্রতিটি স্তর header যোগ করে — *encapsulation*), প্রাপকের দিকে নিচ থেকে উপরে ওঠে (*decapsulation*)।

স্তর	নাম	প্রধান কাজ	PDU	প্রোটোকল / ডিভাইস
7	Application	ব্যবহারকারী/অ্যাপ্লিকেশনকে নেটওয়ার্ক সেবা দেয় (ওয়েব, ই-মেইল, ফাইল ট্রান্সফার)	Data	HTTP, HTTPS, FTP, SMTP, DNS
6	Presentation	ডেটার রূপান্তর (syntax), encryption/decryption, compression	Data	TLS/SSL, JPEG, ASCII, MPEG
5	Session	সেশন স্থাপন, রক্ষণাবেক্ষণ ও সমাপ্তি; dialog control, synchronization (checkpoint)	Data	NetBIOS, RPC

4	Transport	end-to-end delivery, segmentation/reassembly, port addressing, flow ও error control	Segment / Datagram	TCP, UDP
3	Network	logical (IP) addressing, routing — উৎস থেকে গন্তব্যে সেরা পথ নির্বাচন	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	বিটকে বৈদ্যুতিক/আলোক/রেডিও সিগন্যালে রূপান্তর; কেবল, কানেক্টর, bit rate	Bit	Cable, Hub, Repeater, NIC

মনে রাখার কৌশল

উপর থেকে নিচে: **A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing। ব্যাংকের উদাহরণ: গ্রাহক ইন্টারনেট ব্যাংকিংয়ে লগইন করলে HTTPS (L7) → TLS এনক্রিপশন (L6) → সেশন (L5) → TCP পোর্ট 443 (L4) → IP রাউটিং (L3) → Ethernet frame (L2) → ফাইবার/তারে বিট (L1)।

প্রশ্ন ৭ BSCS AP 03.01.2026

Compare TCP and UDP protocols with examples. (10)

TCP (Transmission Control Protocol) ও UDP (User Datagram Protocol) দুটোই OSI-র Transport layer প্রোটোকল। TCP নির্ভরযোগ্যতা দেয়, UDP দেয় গতি।

ভিত্তি	TCP	UDP
সংযোগ	Connection-oriented (3-way handshake: SYN → SYN-ACK → ACK)	Connectionless — সরাসরি পাঠায়
নির্ভরযোগ্যতা	Reliable — ACK, retransmission, sequence number	Unreliable — ডেলিভারির নিশ্চয়তা নেই
ক্রম	In-order delivery	ক্রম নিশ্চিত নয়
Flow/Congestion control	আছে (sliding window, slow start)	নেই
Header	20–60 byte	8 byte
গতি/overhead	ধীর, বেশি overhead	দ্রুত, কম overhead
Broadcast/Multicast	সমর্থন করে না	সমর্থন করে
উদাহরণ	HTTP/HTTPS (80/443), FTP (20/21), SMTP (25), SSH (22), ইন্টারনেট ব্যাংকিং লেনদেন	DNS (53), DHCP (67/68), SNMP (161), VoIP, লাইভ ভিডিও স্ট্রিমিং, অনলাইন গেম

উপসংহার: যেখানে প্রতিটি বাইট সঠিকভাবে পৌঁছানো জরুরি (fund transfer, file download), সেখানে TCP; যেখানে সামান্য ডেটা হারালেও সমস্যা নেই কিন্তু দেরি অগ্রহণযোগ্য (ভয়েস/ভিডিও কল), সেখানে UDP।

প্রশ্ন ৮ BSCS AP 03.01.2026

Explain the difference between compiler and interpreter. (10)

দুটোই high-level ভাষায় লেখা source code-কে মেশিন বুঝতে পারে এমন রূপে রূপান্তর করে। **Compiler** পুরো প্রোগ্রাম একবারে অনুবাদ করে একটি executable (object code) তৈরি করে; **Interpreter** প্রতিটি statement একে একে অনুবাদ

করে সঙ্গে সঙ্গে চালায়।

ভিত্তি	Compiler	Interpreter
অনুবাদের একক	পুরো প্রোগ্রাম একসাথে	লাইন বাই লাইন (statement by statement)
আউটপুট	আলাদা object/executable ফাইল (.exe, .o)	কোনো স্থায়ী executable নেই
এক্সিকিউশন গতি	দ্রুত (একবার কম্পাইল, বারবার চালানো)	ধীর (প্রতিবার অনুবাদ)
Error রিপোর্ট	কম্পাইলের শেষে সব error একসাথে	প্রথম error-এ থেমে যায় — debugging সহজ
মেমরি	object code রাখতে বেশি	কম
Source code প্রয়োজন	চালানোর সময় লাগে না	প্রতিবার লাগে
উদাহরণ	C, C++, Go, Rust	Python, JavaScript, Ruby, PHP

Hybrid পদ্ধতি: Java-তে `javac` সোর্সকে bytecode-এ কম্পাইল করে, তারপর JVM সেটি interpret করে বা JIT (Just-In-Time) compiler দিয়ে native code-এ রূপান্তর করে। Python-ও প্রথমে .pyc bytecode বানায়। Compiler-এর ধাপ: Lexical analysis → Syntax analysis → Semantic analysis → Intermediate code → Optimization → Code generation।

> **প্রশ্ন ৯** BSCS AP 03.01.2026
>
> *What is cloud computing? Mention its service models. (10)*

সংজ্ঞা (NIST): Cloud computing হলো এমন একটি মডেল যেখানে নেটওয়ার্কের মাধ্যমে কনফিগারযোগ্য computing resource-এর (server, storage, network, application, service) একটি shared pool-এ চাহিদামতো প্রবেশাধিকার পাওয়া যায়, যা ন্যূনতম ব্যবস্থাপনা প্রচেষ্টায় দ্রুত দেওয়া ও ফেরত নেওয়া যায়।

৫টি মূল বৈশিষ্ট্য: On-demand self-service, Broad network access, Resource pooling (multi-tenancy), Rapid elasticity, Measured service (pay-as-you-go)।

Service Model	প্রোভাইডার যা দেয়	গ্রাহক যা নিয়ন্ত্রণ করে	উদাহরণ
IaaS (Infrastructure as a Service)	Virtual machine, storage, network	OS, runtime, অ্যাপ্লিকেশন, ডেটা	AWS EC2, Azure VM, Google Compute Engine
PaaS (Platform as a Service)	IaaS + OS + runtime + middleware + database	শুধু অ্যাপ্লিকেশন ও ডেটা	Google App Engine, Heroku, Azure App Service
SaaS (Software as a Service)	সম্পূর্ণ তৈরি সফটওয়্যার	শুধু ব্যবহার ও কিছু সেটিংস	Gmail, Microsoft 365, Salesforce, Zoom

Deployment model: Public, Private, Hybrid, Community cloud। ব্যাংকের ক্ষেত্রে সংবেদনশীল গ্রাহক-ডেটার জন্য সাধারণত private/hybrid cloud এবং নিয়ন্ত্রক সংস্থার data-localization নির্দেশনা মেনে চলতে হয়।

সুবিধা: মূলধনী ব্যয় কম (CapEx → OpEx), দ্রুত scale up/down, high availability, disaster recovery সহজ। **ঝুঁকি:** ডেটা নিরাপত্তা ও গোপনীয়তা, vendor lock-in, ইন্টারনেট নির্ভরতা।

> **প্রশ্ন ১০** BSCS AP 03.01.2026
>
> *What is the difference between functional and non-functional requirements? What is requirement validation? (10)*

ভিত্তি	Functional Requirement	Non-functional Requirement
সংজ্ঞা	সিস্টেম **কী করবে** — নির্দিষ্ট আচরণ/ফিচার	সিস্টেম **কতটা ভালোভাবে** করবে — গুণগত বৈশিষ্ট্য ও সীমাবদ্ধতা
ফোকাস	Input, processing, output, business rule	Performance, security, usability, reliability, scalability
যাচাই	Functional/black-box testing	Performance, load, security testing
না থাকলে	সিস্টেম কাজই করবে না	সিস্টেম চলবে কিন্তু অসন্তোষজনকভাবে
ব্যাংকিং উদাহরণ	"গ্রাহক নিজের দুই হিসাবের মধ্যে fund transfer করতে পারবে"; "দৈনিক লেনদেনের স্টেটমেন্ট PDF-এ ডাউনলোড করা যাবে"	"প্রতিটি লেনদেন ২ সেকেন্ডের মধ্যে সম্পন্ন হবে"; "সিস্টেম ৯৯.৯% সময় চালু থাকবে"; "পাসওয়ার্ড hash করে সংরক্ষণ হবে"

Requirement Validation: নথিভুক্ত requirement (SRS) সত্যিই গ্রাহক যে সিস্টেম চান সেটিকে সঠিকভাবে বর্ণনা করে কিনা তা যাচাইয়ের প্রক্রিয়া — অর্থাৎ "*Are we building the right system?*"। ডেভেলপমেন্টের পর requirement-এর ভুল ঠিক করা অনেক ব্যয়বহুল, তাই আগেই যাচাই করা হয়।

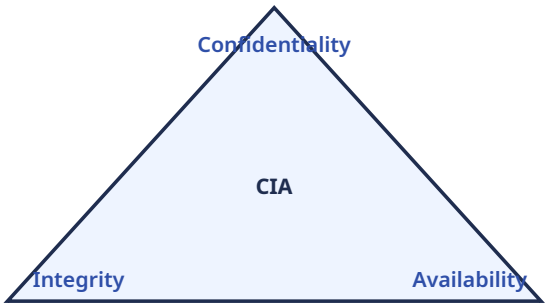
- **যা যাচাই করা হয়:** Validity (সত্যিই দরকারি কিনা), Consistency (পরস্পরবিরোধী নয়), Completeness (সব ফাংশন ও সীমাবদ্ধতা আছে), Realism (বাজেট ও প্রযুক্তিতে সম্ভব), Verifiability (পরীক্ষা করে প্রমাণযোগ্য)।
- **কৌশল:** Requirement review/inspection (স্টেকহোল্ডারদের সাথে), Prototyping, Test-case generation (টেস্ট লেখা না গেলে requirement অস্পষ্ট), Traceability matrix।

> **প্রশ্ন ১১** BSCS AP 03.01.2026
>
> *What is authentication and authorization? What is the CIA triad in cyber security? (10)*

ভিত্তি	Authentication	Authorization
প্রশ্ন	"তুমি কে?" — পরিচয় যাচাই	"তুমি কী করতে পারবে?" — অনুমতি নির্ধারণ
ক্রম	আগে হয়	Authentication-এর পরে হয়
পদ্ধতি	Password, OTP, biometric, smart card, MFA	Role-based access (RBAC), ACL, policy
ব্যবহারকারী দেখে/বদলায়	হ্যাঁ (পাসওয়ার্ড বদলাতে পারে)	না — admin নির্ধারণ করে
ব্যাংক উদাহরণ	টেলার user ID + password + OTP দিয়ে CBS-এ লগইন	টেলার ৫ লাখ টাকা পর্যন্ত অনুমোদন দিতে পারে, এর বেশি হলে ম্যানেজারের অনুমোদন লাগে

CIA Triad — তথ্য নিরাপত্তার তিনটি মূল লক্ষ্য:

- **Confidentiality (গোপনীয়তা):** শুধু অনুমোদিত ব্যক্তি তথ্য দেখতে পারবে। উপায়: encryption, access control, MFA। উদাহরণ: গ্রাহকের হিসাবের ব্যালান্স অন্য কেউ দেখতে পারবে না।
- **Integrity (অখণ্ডতা):** তথ্য অননুমোদিতভাবে পরিবর্তন হবে না। উপায়: hashing, digital signature, checksum, audit log। উদাহরণ: ট্রান্সফারের পথে টাকার অংক বদলানো যাবে না।
- **Availability (প্রাপ্যতা):** প্রয়োজনের সময় অনুমোদিত ব্যবহারকারী তথ্য/সেবা পাবে। উপায়: redundancy, backup, DR site, DDoS protection, UPS। উদাহরণ: ATM ও ইন্টারনেট ব্যাংকিং ২৪/৭ চালু থাকবে।



চিত্র ২.১: CIA Triad

প্রশ্ন ১২ BSCS AP 03.01.2026

What is social engineering? What is hashing? How is it different from encryption? What is vulnerability assessment? (10)

(ক) Social Engineering: প্রযুক্তিগত দুর্বলতার বদলে মানুষের বিশ্বাস, ভয়, তাড়াহুড়া বা কৌতূহলকে কাজে লাগিয়ে গোপন তথ্য (পাসওয়ার্ড, OTP, কার্ড নম্বর) আদায় বা ক্ষতিকর কাজ করানো। প্রকারভেদ: *Phishing* (ভুয়া ই-মেইল/ওয়েবসাইট), *Vishing* (ফোনে "ব্যাংক থেকে বলছি"), *Smishing* (SMS), *Pretexting* (মিথ্যা পরিচয়/গল্প), *Baiting* (সংক্রমিত USB ফেলে রাখা), *Tailgating* (কর্মীর পিছে পিছে সুরক্ষিত কক্ষে ঢোকা)। প্রতিরোধ: সচেতনতা প্রশিক্ষণ, "ব্যাংক কখনো OTP চায় না" বার্তা, MFA, ই-মেইল ফিল্টারিং।

(খ) Hashing: যেকোনো দৈর্ঘ্যের input থেকে একটি নির্দিষ্ট দৈর্ঘ্যের output (hash/digest) তৈরি করে এমন **একমুখী (one-way)** গাণিতিক ফাংশন — যেমন SHA-256 সবসময় 256-bit আউটপুট দেয়। ভালো hash function-এর বৈশিষ্ট্য: deterministic, দ্রুত, pre-image resistant (hash থেকে input বের করা যায় না), collision resistant, avalanche effect (input-এ এক বিট বদলালে hash সম্পূর্ণ বদলে যায়)।

ভিত্তি	Hashing	Encryption
দিক	একমুখী — ফেরত আনা যায় না	দ্বিমুখী — key দিয়ে decrypt করা যায়
Key	লাগে না (HMAC-এ লাগে)	লাগে (symmetric বা public/private)
আউটপুট দৈর্ঘ্য	নির্দিষ্ট	সাধারণত input-এর সমানুপাতিক
উদ্দেশ্য	Integrity, পাসওয়ার্ড সংরক্ষণ	Confidentiality
উদাহরণ	SHA-256, SHA-3, bcrypt (MD5/SHA-1 এখন দুর্বল)	AES, RSA, 3DES
ব্যাংক ব্যবহার	ব্যবহারকারীর পাসওয়ার্ড salt + hash করে ডাটাবেসে রাখা	গ্রাহকের কার্ড/হিসাবের তথ্য AES দিয়ে এনক্রিপ্ট করে রাখা ও TLS দিয়ে পাঠানো

(গ) Vulnerability Assessment: সিস্টেম, নেটওয়ার্ক ও অ্যাপ্লিকেশনের নিরাপত্তা দুর্বলতা পদ্ধতিগতভাবে **খুঁজে বের করা, পরিমাপ করা ও অগ্রাধিকার দেওয়া**র প্রক্রিয়া। ধাপ: (১) Asset তালিকা ও scope নির্ধারণ, (২) স্বয়ংক্রিয় স্ক্যান (Nessus, OpenVAS, Qualys) ও manual review, (৩) CVSS স্কোর দিয়ে ঝুঁকি মূল্যায়ন, (৪) রিপোর্ট ও remediation (patch, configuration fix), (৫) পুনরায় স্ক্যান। *Penetration testing*-এর সাথে পার্থক্য: VA দুর্বলতা খোঁজে ও তালিকা করে; pen-test সেই দুর্বলতা কাজে লাগিয়ে বাস্তবে ঢোকা যায় কিনা প্রমাণ করে।

প্রশ্ন ১৩ BSCS AP 03.01.2026

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

Waterfall model একটি linear-sequential SDLC মডেল: Requirement → Design → Implementation → Testing → Deployment → Maintenance; প্রতিটি ধাপ সম্পূর্ণ শেষ হলে তবেই পরের ধাপ শুরু হয়। স্থিতিশীল ও সুস্পষ্ট requirement-এর ছোট প্রকল্পে এটি ভালো, কিন্তু আধুনিক পরিবর্তনশীল প্রকল্পে এর কয়েকটি গুরুতর সীমাবদ্ধতা আছে:

Waterfall-এর সীমাবদ্ধতা	Agile কীভাবে সমাধান করে
শুরুতেই সব requirement স্থির ধরে নেয়; পরে পরিবর্তন খুব ব্যয়বহুল	Requirement পরিবর্তনকে স্বাগত জানায় — প্রতিটি sprint-এর আগে product backlog পুনর্বিন্যাস
Working software শুধু প্রকল্পের শেষে পাওয়া যায়	প্রতি ১–৪ সপ্তাহের sprint-এ চালানোযোগ্য increment
Testing শেষের দিকে — ত্রুটি দেরিতে ধরা পড়ে, ঠিক করতে খরচ বেশি	প্রতিটি sprint-এ continuous testing, TDD, CI/CD
গ্রাহক শুধু শুরুতে ও শেষে জড়িত — ভুল বোঝাবুঝি শেষে ধরা পড়ে	Product Owner ও গ্রাহক নিয়মিত sprint review-তে ফিডব্যাক দেন
উচ্চ ঝুঁকি — একটি ধাপের ভুল পরের সব ধাপে ছড়ায়	ছোট iteration-এ ঝুঁকি ভাগ হয়ে যায়; retrospective-এ প্রক্রিয়া উন্নতি
অতিরিক্ত ডকুমেন্টেশন-নির্ভর	"Working software over comprehensive documentation" (Agile Manifesto)

সমালোচনামূলক বিশ্লেষণ: Agile সব ক্ষেত্রে শ্রেষ্ঠ নয় — বড় নিয়ন্ত্রিত প্রকল্পে (যেমন নিয়ন্ত্রক রিপোর্টিং সিস্টেম, যেখানে requirement আইনে নির্দিষ্ট) Waterfall-এর পূর্বানুমেয় বাজেট ও ডকুমেন্টেশন সুবিধাজনক। Agile-এর জন্য দক্ষ, স্ব-সংগঠিত দল ও গ্রাহকের সক্রিয় অংশগ্রহণ লাগে; নইলে scope creep হয়। তাই অনেক ব্যাংক "hybrid" পদ্ধতি নেয় — পরিকল্পনা ও কমপ্লায়েন্স Waterfall-এ, ডেভেলপমেন্ট Scrum-এ।

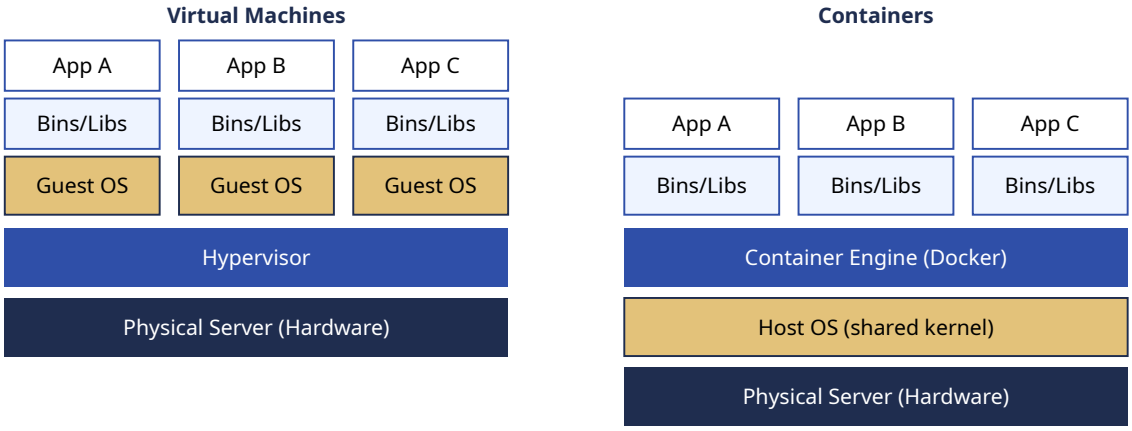
প্রশ্ন ১৪ BSCS AP 03.01.2026

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

Virtualization: একটি ভৌত (physical) হার্ডওয়্যারের রিসোর্স (CPU, RAM, storage, network) সফটওয়্যারের মাধ্যমে ভাগ করে একাধিক বিচ্ছিন্ন ভার্চুয়াল পরিবেশ তৈরি করার প্রযুক্তি, যাতে প্রতিটি পরিবেশ আলাদা কম্পিউটারের মতো আচরণ করে। সুবিধা: সার্ভার consolidation (খরচ ও বিদ্যুৎ সাশ্রয়), দ্রুত provisioning, isolation, snapshot ও সহজ disaster recovery।

Hypervisor (Virtual Machine Monitor): যে সফটওয়্যার স্তর VM তৈরি ও চালায়, প্রতিটি VM-কে হার্ডওয়্যার রিসোর্স বরাদ্দ করে, VM-গুলোকে একে অপর থেকে বিচ্ছিন্ন রাখে এবং privileged instruction নিয়ন্ত্রণ করে।

- **Type 1 (Bare-metal):** সরাসরি হার্ডওয়্যারের উপর চলে — VMware ESXi, Microsoft Hyper-V, KVM, Xen। ডেটা সেন্টারে ব্যবহৃত; দ্রুত ও নিরাপদ।
- **Type 2 (Hosted):** একটি host OS-এর উপর অ্যাপ্লিকেশন হিসেবে চলে — Oracle VirtualBox, VMware Workstation। ডেভেলপমেন্ট/টেস্টিংয়ে ব্যবহৃত।



চিত্র ২.২: VM-এ প্রতিটির নিজস্ব Guest OS; Container-এ সবাই Host OS-এর kernel ভাগ করে

ভিত্তি	Virtual Machine	Container
Virtualization স্তর	হার্ডওয়্যার স্তর (hypervisor)	OS স্তর (namespaces, cgroups)
OS	প্রতিটিতে সম্পূর্ণ Guest OS	Host OS-এর kernel শেয়ার করে
আকার	GB পর্যায়ে	MB পর্যায়ে
চালু হওয়ার সময়	মিনিট	সেকেন্ড বা তার কম
Isolation	শক্তিশালী (আলাদা kernel)	তুলনামূলক দুর্বল (kernel শেয়ার্ড)
ভিন্ন OS চালানো	সম্ভব (Linux host-এ Windows VM)	Host kernel-এর সাথে সামঞ্জস্যপূর্ণ হতে হয়
উদাহরণ	VMware, Hyper-V, KVM VM	Docker, Podman; orchestration: Kubernetes
উপযুক্ত	Legacy অ্যাপ, ভিন্ন OS, কঠোর isolation	Microservices, CI/CD, দ্রুত scaling

প্রশ্ন ১৫ 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)

ব্যাংকে user account তৈরি একটি নিয়ন্ত্রিত প্রক্রিয়া (User Access Management), যাতে প্রত্যেক ব্যবহারকারী শুধু তার কাজের জন্য প্রয়োজনীয় অনুমতি পায় এবং প্রতিটি কাজের জবাবদিহিতা থাকে।

1. **অনুরোধ ও অনুমোদন:** HR যোগদান নিশ্চিত করে; শাখা/বিভাগীয় প্রধান নির্ধারিত User Access Request Form-এ কোন সিস্টেম (CBS, ই-মেইল, ইন্টারনেট ব্যাংকিং ব্যাক-অফিস) ও কোন role লাগবে তা উল্লেখ করে স্বাক্ষর করেন।
2. **পরিচয় যাচাই:** Employee ID, NID ও নিয়োগপত্রের সাথে মিলিয়ে নিশ্চিত হওয়া।
3. **Unique user ID:** নির্দিষ্ট naming convention (যেমন employee ID ভিত্তিক); কোনো shared/generic account নয় — যাতে প্রতিটি কাজ ব্যক্তির সাথে যুক্ত থাকে (accountability)।
4. **Role-Based Access Control (RBAC) ও Least Privilege:** পদ অনুযায়ী পূর্বনির্ধারিত role (টেলার, অফিসার, ম্যানেজার) দেওয়া; প্রয়োজনের অতিরিক্ত অনুমতি নয়। **Segregation of Duties** — একই ব্যক্তি লেনদেন তৈরি ও অনুমোদন (maker-checker) দুটোই করতে পারবে না।
5. **Maker-Checker-এ account তৈরি:** IT admin account তৈরি করেন (maker), আরেকজন যাচাই করে সক্রিয় করেন (checker)।
6. **পাসওয়ার্ড নীতি:** সাময়িক পাসওয়ার্ড নিরাপদ চ্যানেলে (সিল করা খাম/আলাদা SMS) দেওয়া এবং **প্রথম লগইনে বাধ্যতামূলক পরিবর্তন**; ন্যূনতম দৈর্ঘ্য (যেমন ৮–১২+ অক্ষর), বড়-ছোট হাতের অক্ষর, সংখ্যা ও বিশেষ চিহ্নের মিশ্রণ; পূর্বের

কয়েকটি পাসওয়ার্ড পুনর্ব্যবহার নিষেধ (password history); নির্ধারিত সময় পর পরিবর্তন (নীতিমালা অনুযায়ী); কয়েকবার ভুল চেষ্টায় account lockout; পাসওয়ার্ড শুধু salted hash আকারে সংরক্ষণ।

7. **Multi-Factor Authentication (MFA):** সংবেদনশীল সিস্টেম ও রিমোট অ্যাক্সেসে OTP/হার্ডওয়্যার টোকেন।
8. **অন্যান্য নিয়ন্ত্রণ:** session timeout, লগইন সময়/টার্মিনাল সীমাবদ্ধতা, নিরাপত্তা সচেতনতা প্রশিক্ষণ ও Acceptable Use Policy-তে স্বাক্ষর।
9. **লগিং ও অডিট:** account তৈরি, role পরিবর্তন ও প্রতিটি লগইন audit log-এ রাখা; নিয়মিত (যেমন ত্রৈমাসিক) access review।
10. **পরিবর্তন ও অফবোর্ডিং:** বদলি/পদোন্নতিতে পুরোনো অনুমতি প্রত্যাহার; চাকরি ছাড়লে বা দীর্ঘ ছুটিতে account সঙ্গে সঙ্গে disable।

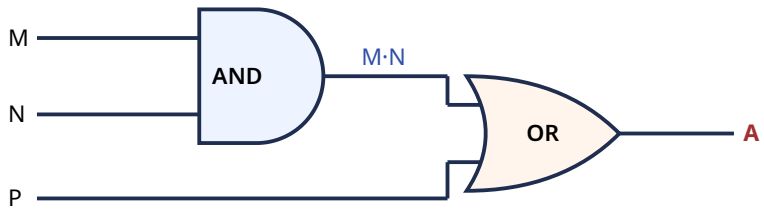
এগুলো বাংলাদেশ ব্যাংকের ICT Security Guideline-এর user access management ধারণার সাথে সামঞ্জস্যপূর্ণ সাধারণ সর্বোত্তম চর্চা; নির্দিষ্ট সংখ্যা (দৈর্ঘ্য, মেয়াদ) প্রতিটি ব্যাংকের নীতিমালা অনুযায়ী।

> **প্রশ্ন ১৬** 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)*

চলক নির্ধারণ: M = Motion detected (1 = হ্যাঁ), N = Night time (1 = রাত), P = Panic button pressed (1 = চাপা)। আউটপুট **A = (M · N) + P**। তিনটি input থাকায় $2^3 = 8$টি সারি।

M	N	P	M·N	A = M·N + P	ব্যাখ্যা
0	0	0	0	**0**	কিছুই ঘটেনি
0	0	1	0	**1**	প্যানিক বাটন
0	1	0	0	**0**	রাত কিন্তু নড়াচড়া নেই
0	1	1	0	**1**	প্যানিক বাটন
1	0	0	0	**0**	দিনে নড়াচড়া — স্বাভাবিক
1	0	1	0	**1**	প্যানিক বাটন
1	1	0	1	**1**	রাতে নড়াচড়া
1	1	1	1	**1**	দুটোই

Canonical SOP: A = Σm(1, 3, 5, 6, 7) = M'N'P + M'NP + MN'P + MNP' + MNP, যা সরল করলে আবার **A = MN + P** হয় (m1,3,5,7 মিলে P; m6,7 মিলে MN)। বাস্তবায়নে একটি 2-input AND gate ও একটি 2-input OR gate লাগে।



চিত্র ২.৩: অ্যালার্ম সার্কিট — A = M·N + P

প্রশ্ন ১৭ 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)

উত্তর: না — User X Folder 1-এ লিখতে (Write) পারবে না, তবে পড়তে (Read) পারবে।

যুক্তি: Windows NTFS-এর মতো ACL-ভিত্তিক সিস্টেমে কোনো ব্যবহারকারীর কার্যকর অনুমতি (effective permission) হলো তার নিজস্ব অনুমতি এবং সে যেসব গ্রুপের সদস্য সেগুলোর অনুমতির সমষ্টি — **তবে কোনো Deny থাকলে তা Allow-কে বাতিল করে**।

উৎস	অনুমতি	ধরন
Group A (X সদস্য)	Allow Read	Explicit Allow
Group B (X সদস্য)	Deny Write	**Explicit Deny**
User X (নিজে)	Allow Write	Explicit Allow
কার্যকর ফল		Read = Allow ✔; Write = Deny ✘

Deny ACE ব্যবহারকারীকে সরাসরি দেওয়া হোক বা তার গ্রুপকে — দুটোই সমানভাবে প্রযোজ্য। একই স্তরে (দুটোই Folder 1-এ explicit) Deny ও Allow থাকলে **Deny জয়ী হয়**, কারণ Windows ACL-এর ACE গুলো canonical order-এ মূল্যায়ন করে এবং প্রথম মিল পাওয়া Deny-তে থেমে যায়।

Precedence (অগ্রাধিকার) ক্রম — উপরেরটি শক্তিশালী

1. **Explicit Deny** (এই অবজেক্টেই সরাসরি সেট করা Deny)
2. **Explicit Allow**
3. **Inherited Deny** (parent folder থেকে উত্তরাধিকার)
4. **Inherited Allow**

লক্ষ করুন: Explicit Allow একটি Inherited Deny-কে হারাতে পারে; কিন্তু Explicit Deny-কে কোনো Allow হারাতে পারে না।

সমাধান (যদি X-কে লিখতে দিতে হয়): X-কে Group B থেকে সরানো, অথবা Group B-এর Deny ACE সরিয়ে প্রয়োজনমতো সীমিত অনুমতি দেওয়া। নিরাপত্তার দৃষ্টিতে Deny কম ব্যবহার করে "least privilege" অনুযায়ী শুধু প্রয়োজনীয় Allow দেওয়াই সর্বোত্তম চর্চা।

প্রশ্ন ১৮ 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 = 4টি ডিস্ক, প্রতিটি S = 2 TB; মোট raw capacity = 4 × 2 = 8 TB।

RAID	সূত্র	হিসাব	Usable	Fault tolerance
RAID 0 (Striping)	N × S	4 × 2	**8 TB**	নেই — একটি ডিস্ক নষ্ট হলে সব ডেটা হারায়
RAID 5 (Striping + distributed parity)	(N − 1) × S	3 × 2	**6 TB**	যেকোনো ১টি ডিস্ক নষ্ট সহ্য করে

RAID 10 (Mirror + Stripe)	(N × S) / 2	8 / 2	**4 TB**	প্রতিটি mirror pair থেকে ১টি করে ডিস্ক নষ্ট সহ্য করে

ব্যাখ্যা: RAID 0 ডেটা ভাগ করে সব ডিস্কে লেখে — সর্বোচ্চ গতি ও ধারণক্ষমতা, কিন্তু কোনো redundancy নেই। RAID 5-এ এক ডিস্কের সমান জায়গা parity-তে যায় (parity সব ডিস্কে ছড়ানো), তাই ৬ TB। RAID 10-এ প্রথমে দুটি করে ডিস্ক mirror (RAID 1) করে তারপর stripe (RAID 0) করা হয় — অর্ধেক জায়গা কপিতে যায়, তবে লেখার গতি ও পুনর্গঠন RAID 5-এর চেয়ে ভালো; তাই ব্যাংকের ডাটাবেস সার্ভারে RAID 10 জনপ্রিয়।

> **প্রশ্ন ১৯** 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)*

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



চিত্র ২.৪: কম্পিউটার বুট প্রক্রিয়ার ক্রম

1. **BIOS/UEFI POST:** পাওয়ার অন হলে ROM/flash-এ থাকা firmware চালু হয় এবং Power-On Self-Test-এ CPU, RAM, কীবোর্ড, ডিস্ক ইত্যাদি হার্ডওয়্যার পরীক্ষা করে; তারপর boot order অনুযায়ী boot device বাছাই করে।
2. **MBR/GPT Lookup:** বুট ডিস্কের partition table পড়া হয় — legacy BIOS-এ প্রথম সেক্টরের MBR (512 byte), UEFI-তে GPT ও EFI System Partition — এবং সেখান থেকে bootloader (GRUB, Windows Boot Manager) চালু হয়।
3. **OS Kernel Load:** Bootloader অপারেটিং সিস্টেমের kernel (ও initramfs) মেমরিতে লোড করে নিয়ন্ত্রণ kernel-কে দেয়।
4. **Driver Initialization:** Kernel হার্ডওয়্যার শনাক্ত করে device driver লোড করে, ফাইল সিস্টেম mount করে এবং system service/daemon (Linux-এ init/systemd) চালু করে।
5. **User Login:** লগইন স্ক্রিন/প্রম্পট আসে; authentication সফল হলে user session শুরু হয়।

> **প্রশ্ন ২০** 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)*

লক্ষ করুন: DEPARTMENTS টেবিলের key কলামের নাম `Dept`, আর EMPLOYEES-এর foreign key `DeptID` — join শর্তে এই দুটি মেলাতে হবে।

লজিক্যাল ধাপ:

1. DEPARTMENTS টেবিল থেকে যে সারিতে `DeptName = 'IT'` সেটি খুঁজে তার `Dept` মান বের করি।
2. EMPLOYEES টেবিলের প্রতিটি সারির `DeptID`-কে ওই `Dept` মানের সাথে মেলাই (JOIN)।
3. মিলে যাওয়া সারিগুলো থেকে শুধু `Name` কলাম প্রদর্শন করি (প্রয়োজনে নাম অনুযায়ী সাজাই)।

```sql
-- পদ্ধতি ১: 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;

-- পদ্ধতি ২: Subquery
SELECT Name
FROM   EMPLOYEES
WHERE  DeptID IN (SELECT Dept FROM DEPARTMENTS WHERE DeptName = 'IT');
```

উদাহরণ: DEPARTMENTS = {(10, 'HR'), (20, 'IT')}; EMPLOYEES = {(1, 'Rahim', 20), (2, 'Karim', 10), (3, 'Nusrat', 20)} হলে ফলাফল: **Nusrat, Rahim**। INNER JOIN-এ যেসব কর্মীর DeptID NULL বা অমিল, তারা বাদ পড়বে — এখানে সেটাই কাম্য।

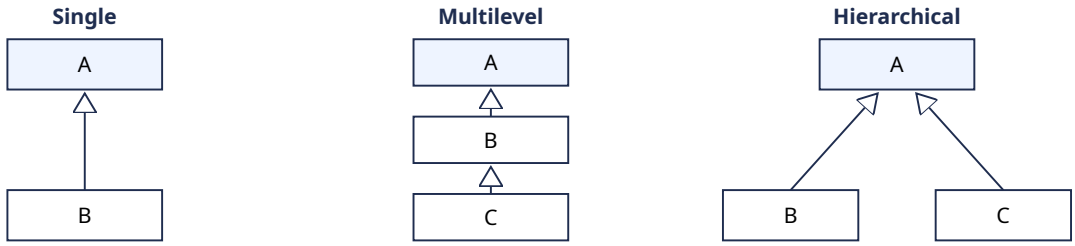
২.২ Paper E — BSCS Various Bank Assistant Programmer, ১৭.০১.২০২৬

বিষয়	তথ্য
পরীক্ষা	BSCS — বিভিন্ন ব্যাংকের Assistant Programmer
তারিখ	১৭/০১/২০২৬
প্রশ্নকর্তা (সংকলন অনুযায়ী)	BIBM/CTI
MCQ	৮০ নম্বর (IT ৭৫ + Math+GK ২৫) — সংকলনের তথ্য
লিখিত (প্রশ্নপত্র অনুযায়ী)	Subject Related প্রশ্ন ১–১৫ (১৫০) + Mathematics প্রশ্ন ১৬–১৯ + English Proficiency প্রশ্ন ২০–২২ (grammar, paragraph, sentence making)
লক্ষণীয়	সংকলনের হেডারে "Focus + B/E" লেখা থাকলেও প্রকৃত প্রশ্নপত্রে ইংরেজি অংশটি "English Proficiency" — যা ৩১ অক্টোবরের (25106) বিজ্ঞপ্তির "English Language Proficiency ৩০"-এর সাথে মেলে

প্রশ্ন ১ 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 (উত্তরাধিকার) হলো OOP-এর এমন একটি বৈশিষ্ট্য যার মাধ্যমে একটি class (child/subclass/derived) অন্য একটি class-এর (parent/superclass/base) attribute ও method অর্জন করে, এবং চাইলে নতুন সদস্য যোগ বা পুরোনো method override করতে পারে। এটি "**IS-A**" সম্পর্ক প্রকাশ করে (SavingsAccount IS-A Account)। সুবিধা: code reusability, সহজ রক্ষণাবেক্ষণ, extensibility এবং runtime polymorphism-এর ভিত্তি।



চিত্র ২.৫: তিন ধরনের inheritance (তীর child থেকে parent-এর দিকে — UML)

1. **Single Inheritance:** একটি child একটি মাত্র parent থেকে উত্তরাধিকার পায়। উদাহরণ: `SavingsAccount extends Account`।
2. **Multilevel Inheritance:** উত্তরাধিকারের শৃঙ্খল — C, B থেকে এবং B, A থেকে। উদাহরণ: `Account → SavingsAccount → StudentSavingsAccount`।
3. **Hierarchical Inheritance:** একটি parent থেকে একাধিক child। উদাহরণ: `Account` থেকে `SavingsAccount` ও `CurrentAccount`।

অন্যান্য: **Multiple** (একাধিক parent — C++ সমর্থন করে; Java-তে class দিয়ে নয়, interface দিয়ে সম্ভব, কারণ diamond problem এড়াতে) এবং **Hybrid** (একাধিক ধরনের মিশ্রণ)।

```
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 (Account-এর দ্বিতীয় child)
```

প্রশ্ন ২ BSCS AP 17.01.2026

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

SDLC হলো একটি সুশৃঙ্খল প্রক্রিয়া কাঠামো, যা পরিকল্পনা থেকে শুরু করে রক্ষণাবেক্ষণ পর্যন্ত সফটওয়্যার তৈরির প্রতিটি ধাপ, তার কাজ ও ডেলিভারেবল সংজ্ঞায়িত করে — যাতে নির্ধারিত সময়, বাজেট ও মানসম্পন্ন সফটওয়্যার তৈরি হয়।



চিত্র ২.৬: SDLC-এর ধাপসমূহ

ধাপ	কাজ	আউটপুট
1. Planning / Feasibility	লক্ষ্য, scope, বাজেট, সময়সূচি; প্রযুক্তিগত, অর্থনৈতিক ও আইনগত সম্ভাব্যতা যাচাই; ঝুঁকি চিহ্নিত	Project plan, feasibility report
2. Requirement Analysis	স্টেকহোল্ডারদের থেকে functional ও non-functional requirement সংগ্রহ ও বিশ্লেষণ	SRS (Software Requirement Specification)
3. Design	System architecture, database (ER), UI ও module design — HLD ও LLD	Design document, ER/UML diagram
4. Implementation (Coding)	ডিজাইন অনুযায়ী কোড লেখা, code review, unit testing	Source code
5. Testing	Integration, system, acceptance (UAT) testing; bug fix	Test report, স্থিতিশীল build
6. Deployment	Production-এ ইনস্টল, ডেটা মাইগ্রেশন, ব্যবহারকারী প্রশিক্ষণ	Live system

7. Maintenance	Corrective (bug), adaptive (নতুন পরিবেশ), perfective (নতুন ফিচার), preventive পরিবর্তন	Updated version

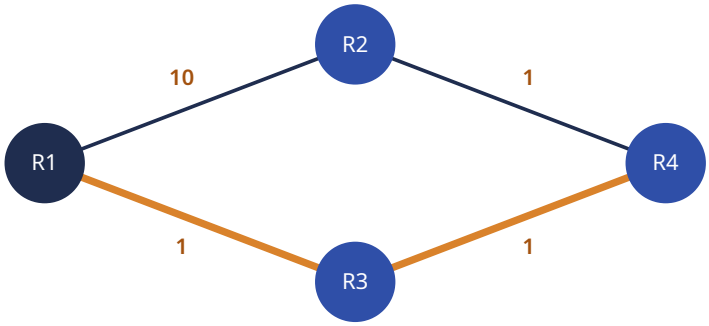
SDLC মডেল: Waterfall, V-Model, Incremental, Spiral, Prototype, Agile (Scrum)। ব্যাংকিং উদাহরণ: নতুন মোবাইল অ্যাপ তৈরিতে UAT ধাপে শাখার কর্মকর্তারা পরীক্ষা করে অনুমোদন দেন, তারপর deployment।

> **প্রশ্ন ৩** 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 প্রতিটি হোস্টকে একটি logical ঠিকানা দেয় (যেমন 192.168.10.25/24), যার *network অংশ* বলে হোস্টটি কোন নেটওয়ার্কে এবং *host অংশ* বলে সেই নেটওয়ার্কের কোন যন্ত্র। রাউটার শুধু network অংশ দেখে প্যাকেট পাঠায়। **Routing** হলো উৎস থেকে গন্তব্য নেটওয়ার্কে পৌঁছানোর পথ নির্ধারণ; রাউটার routing table-এ *longest prefix match* খুঁজে next hop ঠিক করে।

OSPF একটি **link-state interior gateway protocol** (IP protocol number 89), যা Dijkstra-র Shortest Path First (SPF) অ্যালগরিদম দিয়ে প্রতিটি গন্তব্যের সর্বনিম্ন খরচের পথ বের করে। Cost = Reference bandwidth / Interface bandwidth (ডিফল্ট reference 100 Mbps)।

1. **Neighbor discovery:** রাউটারগুলো Hello প্যাকেট (multicast 224.0.0.5) পাঠিয়ে প্রতিবেশী খুঁজে adjacency তৈরি করে; broadcast নেটওয়ার্কে DR/BDR নির্বাচিত হয়।
2. **LSA flooding:** প্রতিটি রাউটার তার নিজস্ব link-এর অবস্থা ও cost নিয়ে **Link-State Advertisement (LSA)** তৈরি করে area-র সব রাউটারে ছড়িয়ে দেয়।
3. **LSDB:** সব LSA মিলে প্রতিটি রাউটারে একই Link-State Database তৈরি হয় — অর্থাৎ পুরো topology-র মানচিত্র।
4. **SPF গণনা:** প্রতিটি রাউটার নিজেকে root ধরে Dijkstra চালিয়ে shortest-path tree বানায় এবং সেরা পথগুলো routing table-এ বসায়।
5. **Packet forwarding (hop-by-hop):** প্যাকেট এলে রাউটার গন্তব্য IP-র সাথে routing table মিলিয়ে next-hop-এ পাঠায়; প্রতিটি হপে TTL ১ কমে; data-link স্তরে নতুন frame (MAC) তৈরি হয় কিন্তু source/destination IP একই থাকে।
6. **পরিবর্তনে দ্রুত অভিযোজন:** কোনো link নষ্ট হলে নতুন LSA ছড়ায়, সবাই SPF পুনরায় গণনা করে — দ্রুত convergence। বড় নেটওয়ার্কে **Area** ব্যবহার করা হয় (Area 0 = backbone; ABR রাউটার area সংযুক্ত করে), যাতে LSA flooding ও SPF গণনা সীমিত থাকে।



চিত্র ২.৭: R1 থেকে R4 — cost সর্বনিম্ন পথ R1→R3→R4 (cost 2), R1→R2→R4 নয় (cost 11)

উদাহরণ: R1-এর SPF (Dijkstra) গণনা

ধাপ	নির্বাচিত নোড	R2	R3	R4
শুরু	R1 (0)	10 (via R1)	1 (via R1)	∞
১	R3 (1)	10	✔ 1	1 + 1 = 2 (via R3)
২	R4 (2)	min(10, 2 + 1) = 3 (via R4)	✔	✔ 2
৩	R2 (3)	✔ 3	✔	✔

ফল: R1-এর routing table-এ R2-গামী প্যাকেটও next-hop R3 দিয়ে যাবে (খরচ 3), সরাসরি link (খরচ 10) নয় — অর্থাৎ OSPF hop সংখ্যা নয়, মোট cost দেখে।

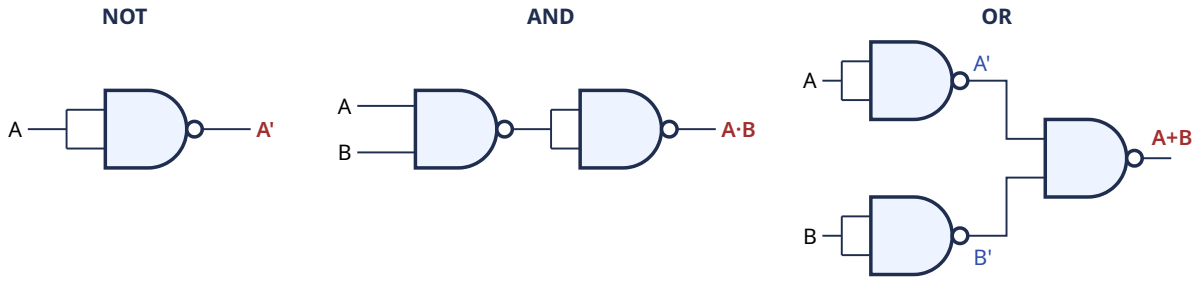
প্রশ্ন ৪ BSCS AP 17.01.2026

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

Universal Gate: যে gate শুধু নিজেকে ব্যবহার করে মৌলিক তিনটি gate (NOT, AND, OR) — এবং ফলে যেকোনো Boolean function — বাস্তবায়ন করতে পারে। NAND ও NOR হলো universal gate। NAND-এর আউটপুট: Y = (A·B)'।

প্রমাণ: NOT, AND ও OR — এই তিনটি functionally complete সেট; তাই এগুলো শুধু NAND দিয়ে বানানো গেলে NAND universal।

1. **NOT:** NAND-এর দুই ইনপুটে একই সংকেত A দিলে: (A·A)' = A' ✔
2. **AND:** প্রথম NAND: (A·B)'; এটিকে দ্বিতীয় NAND দিয়ে উল্টালে: ((A·B)')' = A·B ✔ (২টি gate)
3. **OR:** De Morgan অনুযায়ী A + B = (A'·B')'। প্রথমে দুটি NAND দিয়ে A' ও B' তৈরি, তারপর তৃতীয় NAND: (A'·B')' = A + B ✔ (৩টি gate)



চিত্র ২.৮: শুধু NAND দিয়ে NOT (১টি), AND (২টি), OR (৩টি gate)

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

শেষ দুই কলাম যথাক্রমে AND ও OR-এর truth table-এর সাথে হুবহু মেলে — তাই প্রমাণিত। (অতিরিক্ত: XOR ৪টি NAND দিয়ে, NOR ৪টি NAND দিয়ে তৈরি করা যায়।) বাস্তবে NAND universal হওয়ায় চিপ নির্মাতারা এক ধরনের gate দিয়েই পুরো সার্কিট বানাতে পারেন — খরচ ও উৎপাদন সহজ; NAND flash memory-ও এই নীতির উপর ভিত্তি করে।

প্রশ্ন ৫ 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)

Process হলো execution-এ থাকা একটি প্রোগ্রাম — যার নিজস্ব address space (code, data, heap, stack), PCB, open file ও অন্যান্য রিসোর্স আছে। **Thread** হলো process-এর ভেতরের execution-এর ক্ষুদ্রতম একক (lightweight process); একই process-এর সব thread code, data, heap ও open file শেয়ার করে, কিন্তু প্রত্যেকের নিজস্ব **program counter, register set ও stack** থাকে।

ভিত্তি	Process	Thread
সংজ্ঞা	Heavyweight, স্বাধীন execution unit	Lightweight, process-এর অংশ
Memory	আলাদা address space	একই process-এর address space শেয়ার
তৈরি ও context switch	ধীর ও ব্যয়বহুল	দ্রুত ও সস্তা
যোগাযোগ	IPC লাগে (pipe, message queue, shared memory)	সরাসরি শেয়ার্ড ভেরিয়েবলে
Isolation	একটি crash হলে অন্যটি নিরাপদ	এক thread-এর ত্রুটি পুরো process-কে ফেলে দিতে পারে
নিজস্ব যা থাকে	PID, PCB, memory, files	Thread ID, PC, registers, stack

Thread ব্যবহারের দুটি সুবিধা:

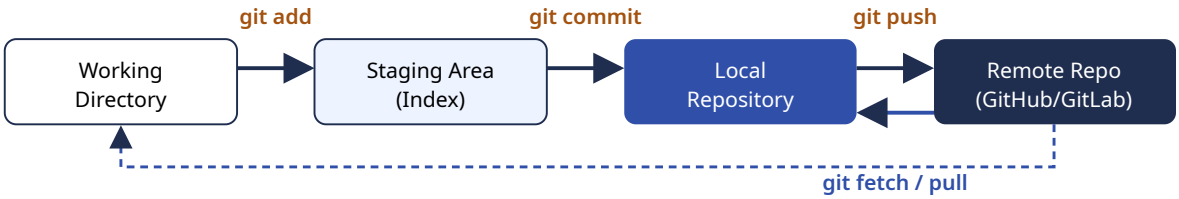
1. **কম overhead ও দ্রুত context switch:** নতুন thread তৈরিতে আলাদা address space বানাতে হয় না, এবং একই process-এর thread-এর মধ্যে switch করতে memory map (TLB/page table) বদলাতে হয় না — তাই process-এর তুলনায় অনেক দ্রুত।
2. **সহজ resource sharing ও যোগাযোগ:** thread-গুলো একই memory শেয়ার করে, তাই IPC ছাড়াই ডেটা আদান-প্রদান করতে পারে।

আরও: *Responsiveness* — একটি thread ব্লক হলেও (যেমন I/O) অন্য thread UI চালু রাখে; *Scalability* — multicore CPU-তে thread-গুলো সমান্তরালে চলে। উদাহরণ: ব্যাংকের ওয়েব সার্ভার প্রতিটি গ্রাহকের অনুরোধ আলাদা thread-এ সামলায়।

প্রশ্ন ৬ 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) হলো এমন সফটওয়্যার যা সময়ের সাথে ফাইল/কোডের প্রতিটি পরিবর্তনের ইতিহাস সংরক্ষণ করে — কে, কখন, কী পরিবর্তন করেছে তা জানা যায়, আগের সংস্করণে ফেরা যায়, এবং একাধিক ডেভেলপার একসাথে কাজ করতে পারে। **Git** একটি *distributed* VCS — প্রত্যেক ডেভেলপারের কাছে পুরো repository-র একটি পূর্ণ কপি (local repo) থাকে; GitHub/GitLab-এ থাকে shared *remote* repo।



চিত্র ২.৯: Git-এর তিন স্থানীয় অঞ্চল ও remote

ভিত্তি	Commit	Push
কাজ	Staged পরিবর্তনের একটি snapshot **local repository**-তে স্থায়ীভাবে সংরক্ষণ	Local repo-র commit গুলো **remote repository**-তে পাঠানো
নেটওয়ার্ক	লাগে না — অফলাইনে সম্ভব	লাগে
দৃশ্যমানতা	শুধু নিজের কম্পিউটারে	দলের সবাই দেখতে/pull করতে পারে
পরিচয়	প্রতিটি commit-এর unique SHA-1 hash, author, সময় ও message থাকে	Remote branch আপডেট হয়; conflict থাকলে আগে pull/merge করতে হয়
কমান্ড	`git commit -m "Fix interest calc"`	`git push origin main`

কমান্ড	কাজ
`git add file`	পরিবর্তন staging area-তে নেওয়া
`git fetch`	Remote-এর নতুন commit নামানো, কিন্তু নিজের branch-এ merge না করা
`git pull`	fetch + merge (remote পরিবর্তন নিজের branch-এ যুক্ত)
`git merge branch`	অন্য branch-এর পরিবর্তন বর্তমান branch-এ যুক্ত করা
`git branch / checkout -b`	নতুন branch তৈরি/পরিবর্তন

> **প্রশ্ন ৭** 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 একটি divide-and-conquer অনুসন্ধান পদ্ধতি: সাজানো array-এর মাঝের উপাদানের সাথে target তুলনা করে প্রতি ধাপে অনুসন্ধানের পরিসর অর্ধেক করে।

1. `low = 0`, `high = n - 1` নির্ধারণ।
2. `low ≤ high` যতক্ষণ: `mid = low + (high - low) / 2`।
3. `A[mid] == target` হলে mid ফেরত (পাওয়া গেছে)।
4. `A[mid] < target` হলে target ডান অর্ধে: `low = mid + 1`।
5. `A[mid] > target` হলে বাম অর্ধে: `high = mid - 1`।
6. Loop শেষে না পেলে −1 (অনুপস্থিত)।

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

ধাপ	low	high	mid	A[mid]	সিদ্ধান্ত
১	0	8	4	42	42 < 89 → low = 5
২	5	8	6	71	71 < 89 → low = 7
৩	7	8	7	89	মিলেছে → index 7 ✔

৯টি উপাদানে মাত্র ৩টি তুলনা; linear search-এ লাগত ৮টি।

```
int binarySearch(int A[], int n, int target) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;   /* overflow এড়াতে */
        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)** (১০ লাখ উপাদানে সর্বোচ্চ ~২০টি তুলনা); Space — O(1) (iterative)।

Unsorted array-এ কেন ব্যর্থ: Binary search-এর মূল অনুমান হলো — A[mid] < target হলে mid-এর বাম দিকের *সব* উপাদানও target-এর চেয়ে ছোট, তাই বাম অর্ধ নিরাপদে বাদ দেওয়া যায়। Unsorted array-তে এই ক্রম-সম্পর্ক নেই, ফলে বাদ দেওয়া অর্ধেই target থাকতে পারে। উদাহরণ: A = [42, 3, 89, 15, 8], target = 3; mid = 2 → A[2] = 89 > 3 → high = 1; mid = 0 → 42 > 3 → high = −1 → "নেই" ফেরত দেয়, যদিও 3 index 1-এ আছে — ক্রম না থাকায় অ্যালগরিদম ভুল অর্ধ বাদ দিয়েছে। তাই unsorted ডেটায় হয় linear search (O(n)), নয়তো আগে sort (O(n log n)) করে তারপর binary search।

প্রশ্ন ৮ BSCS AP 17.01.2026

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

Stack LIFO, Queue FIFO। দুটি stack ব্যবহার করে ক্রম দুবার উল্টালে মূল ক্রম (FIFO) ফিরে আসে। `S1` = inbox (enqueue-এর জন্য), `S2` = outbox (dequeue-এর জন্য)।

- **Enqueue(x):** সবসময় `S1.push(x)` — O(1)।
- **Dequeue():** (১) `S2` খালি না হলে `S2.pop()` ফেরত দাও। (২) `S2` খালি হলে `S1`-এর সব উপাদান একে একে pop করে `S2`-তে push করো (এতে ক্রম উল্টে যায় — সবচেয়ে পুরোনো উপাদান S2-এর মাথায় আসে), তারপর `S2.pop()`। (৩) দুটোই খালি হলে Queue Underflow।

উদাহরণ ট্রেস

অপারেশন	S1 (নিচ → উপর)	S2 (নিচ → উপর)	আউটপুট
enqueue 10, 20, 30	10, 20, 30	—	
dequeue (S2 খালি → স্থানান্তর)	—	30, 20, 10 → pop	**10**
enqueue 40	40	30, 20	
dequeue	40	30	**20**
dequeue	40	—	**30**
dequeue (স্থানান্তর)	—	—	**40**

আউটপুট ক্রম 10, 20, 30, 40 — ঠিক যে ক্রমে ঢুকেছে (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());  // ক্রম উল্টানো
        }
        if (s2.isEmpty()) throw new RuntimeException("Queue Underflow");
        return s2.pop();
    }
}
```

Complexity: Enqueue O(1)। Dequeue worst-case O(n) (স্থানান্তরের সময়), কিন্তু প্রতিটি উপাদান জীবনে সর্বোচ্চ একবার S1 থেকে S2-তে যায় — তাই **amortized O(1)**। (বিকল্প পদ্ধতি: enqueue-কে ব্যয়বহুল করা — প্রতিবার সব উপাদান সরিয়ে নতুনটি নিচে বসানো; তখন enqueue O(n), dequeue O(1)।)

প্রশ্ন ৯ BSCS AP 17.01.2026

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

Bubble Sort-এর লজিক: বারবার পাশাপাশি দুটি উপাদান তুলনা করে ভুল ক্রমে থাকলে অদলবদল (swap) করা হয়। প্রতিটি pass-এর শেষে অবশিষ্ট অংশের সবচেয়ে বড় উপাদান "বুদবুদের মতো" শেষ অবস্থানে উঠে যায়। n উপাদানে সর্বোচ্চ n − 1টি pass লাগে; কোনো pass-এ একটিও swap না হলে array sorted — তখন আগেই থামানো যায় (optimized)।

উদাহরণ: [5, 1, 4, 2, 8]

Pass	তুলনা ও swap	Pass শেষে
১	(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) ঠিক	1 4 2 5 **8**
২	(1,4) ঠিক; (4,2) swap → 1 2 4 5 8; (4,5) ঠিক	1 2 4 **5 8**
৩	(1,2), (2,4) — কোনো swap নেই → থামো	**1 2 4 5 8** ✔

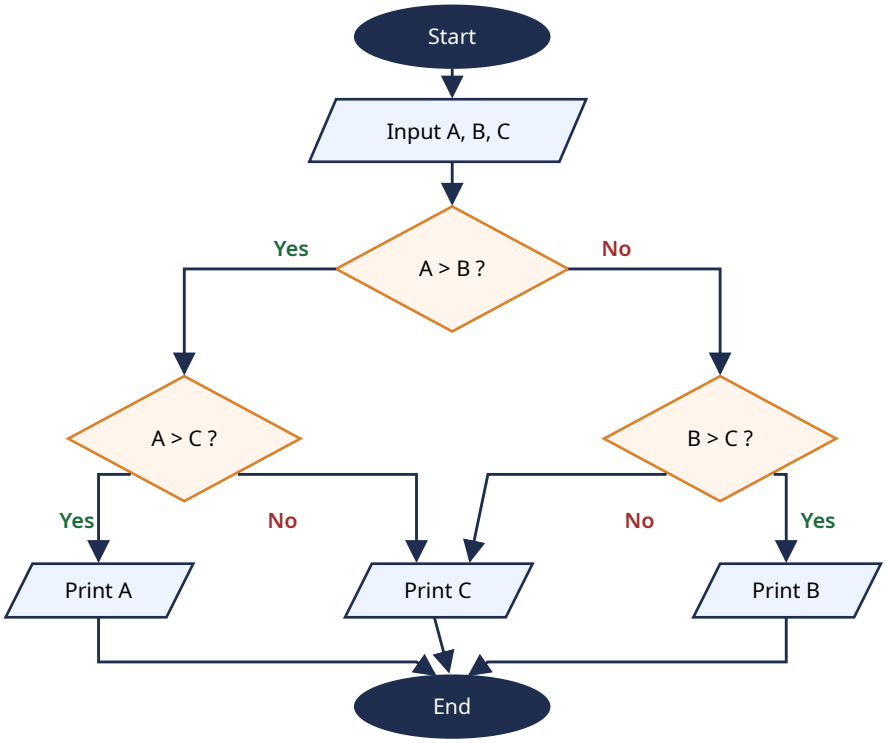
```
void bubbleSort(int a[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int swapped = 0;
        for (int j = 0; j < n - 1 - i; j++) {
            if (a[j] > a[j + 1]) {
                int t = a[j]; a[j] = a[j + 1]; a[j + 1] = t;
                swapped = 1;
            }
        }
    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

ভিত্তি	Bubble Sort	Merge Sort
Best	O(n) (optimized, আগেই sorted)	O(n log n)
Average / Worst	**$O(n^2)$**	**O(n log n)**
Space	O(1) — in-place	O(n) — অতিরিক্ত array
Stable	হ্যাঁ	হ্যাঁ
কৌশল	পাশাপাশি তুলনা ও swap	Divide & conquer: অর্ধেক করে sort, তারপর merge

বড় ডেটায় অদক্ষ কেন: Bubble sort-এ তুলনার সংখ্যা প্রায় $n(n-1)/2$। $n = 10^6$ হলে প্রায় 5×10^{11} তুলনা, আর merge sort-এ $n \log_2 n \approx 10^6 \times 20 = 2 \times 10^7$ — অর্থাৎ প্রায় ২৫,০০০ গুণ কম কাজ। তাছাড়া bubble sort-এ swap-এর সংখ্যাও $O(n^2)$, যা মেমরিতে অনেক লেখার কাজ করে। ফলে ছোট বা প্রায়-সাজানো ডেটায় চলে, কিন্তু লাখো লেনদেনের রেকর্ড সাজাতে merge sort (বা quick sort/Timsort) ব্যবহৃত হয়।

প্রশ্ন ১০ BSCS AP 17.01.2026

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



চিত্র ২.১০: তিনটি ভিন্ন সংখ্যার মধ্যে বৃহত্তম নির্ণয়ের ফ্লোচার্ট

ব্যাখ্যা (অ্যালগরিদম): ধাপ ১: শুরু। ধাপ ২: A, B, C ইনপুট। ধাপ ৩: A > B হলে ধাপ ৪, নইলে ধাপ ৫। ধাপ ৪: A > C হলে A বৃহত্তম, নইলে C বৃহত্তম। ধাপ ৫: B > C হলে B বৃহত্তম, নইলে C বৃহত্তম। ধাপ ৬: ফল প্রিন্ট, শেষ। (সংখ্যাগুলো distinct, তাই সমতার ক্ষেত্র বিবেচনা লাগে না।) প্রতীক: ডিম্বাকৃতি = Start/End, সামান্তরিক = Input/Output, রম্বস = Decision।

```c
#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;
}
```

প্রশ্ন ১১ BSCS AP 17.01.2026

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

সমস্যা (Bounded-Buffer Problem): একটি নির্দিষ্ট আকারের (N স্লট) shared buffer আছে। **Producer** process/thread ডেটা তৈরি করে buffer-এ রাখে, **Consumer** buffer থেকে ডেটা নিয়ে ব্যবহার করে। দুজন একসাথে চলে বলে তিনটি বিষয় নিশ্চিত করতে হয়:

1. **Mutual exclusion:** একই সময়ে দুজন buffer/count পরিবর্তন করলে *race condition* হয় (যেমন `count++` ও `count--` একসাথে চললে count ভুল হয়)। তাই buffer-এ প্রবেশ (critical section) একসাথে একজনই করবে।
2. **Buffer full হলে Producer অপেক্ষা করবে** (overflow রোধ)।
3. **Buffer empty হলে Consumer অপেক্ষা করবে** (underflow রোধ)।

Semaphore দিয়ে সমাধান: `mutex = 1` (binary — পারস্পরিক বর্জন), `empty = N` (খালি স্লট গণনা), `full = 0` (ভরা স্লট গণনা)।

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

Producer:                          Consumer:
while (true) {                     while (true) {
    item = produce();                  wait(full);     // ভরা স্লট না থাকলে ঘুম
    wait(empty);   // জায়গা না থাকলে ঘুম   wait(mutex);
    wait(mutex);                       item = buffer[out];
    buffer[in] = item;                 out = (out + 1) % N;
    in = (in + 1) % N;                 signal(mutex);
    signal(mutex);                     signal(empty);  // একটি স্লট খালি হলো
    signal(full);  // একটি স্লট ভরা হলো    consume(item);
}                                  }
```

সতর্কতা: wait-এর ক্রম গুরুত্বপূর্ণ

Producer যদি আগে `wait(mutex)` করে তারপর `wait(empty)` করে এবং buffer full থাকে, তবে সে mutex ধরে রেখেই ঘুমিয়ে পড়বে; Consumer mutex না পেয়ে আটকে যাবে — **deadlock**। তাই সবসময় counting semaphore (empty/full) আগে, mutex পরে।

বাস্তব উদাহরণ: ATM লেনদেনের অনুরোধ (producer) একটি message queue-তে জমা হয় এবং CBS-এর worker thread (consumer) সেগুলো প্রক্রিয়া করে; print spooler; Kafka-র মতো message broker।

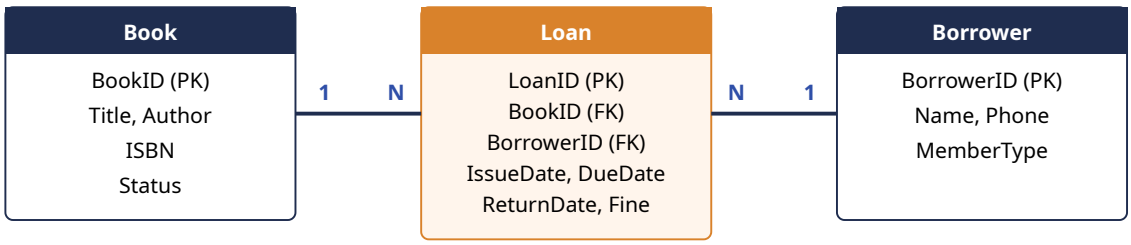
প্রশ্ন ১২ BSCS AP 17.01.2026

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)

উত্তর: বাস্তব লাইব্রেরি ডাটাবেসে Book ও Borrower-এর সম্পর্ক Many-to-Many (M:N)।

- **একজন Borrower অনেকগুলো Book** ধার নিতে পারেন (একসাথে বা সময়ের সাথে)।
- **একটি Book** (একটি কপিও) সময়ের সাথে **অনেক Borrower** ধার নিতে পারেন — লাইব্রেরিকে এই ধারের ইতিহাস (কে কবে নিয়েছিল, কবে ফেরত দিয়েছে, জরিমানা) সংরক্ষণ করতে হয়।

Relational ডাটাবেসে M:N সরাসরি রাখা যায় না; তাই একটি **associative (junction) টেবিল** `Loan` দিয়ে দুটি 1:N সম্পর্কে ভাঙা হয়:



চিত্র ২.১১: Book (1) — (N) Loan (N) — (1) Borrower ⇒ Book ও Borrower-এর মধ্যে M:N

```
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 = এখনো ফেরত দেয়নি
);
```

বিকল্প ব্যাখ্যা (1:N): যদি শুধু *বর্তমান* অবস্থা রাখা হয় (ইতিহাস নয়) এবং Book মানে একটি নির্দিষ্ট ভৌত কপি — তাহলে একটি কপি একসময়ে কেবল একজনের কাছে থাকে, কিন্তু একজন অনেক কপি রাখতে পারেন; তখন Borrower (1) — (N) Book, এবং Book টেবিলে `BorrowerID` foreign key রাখলেই চলে। তবে এতে ধারের ইতিহাস, জরিমানা হিসাব ও রিপোর্ট সম্ভব নয় — তাই ডিজাইনে M:N (Loan টেবিলসহ) বেছে নেওয়াই সঠিক। 1:1 এখানে অযৌক্তিক, কারণ একজন সদস্য শুধু একটি বই নিতে পারবেন এমন নিয়ম বাস্তবে নেই।

> **প্রশ্ন ১৩** 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: লক্ষ্য কোথায় সে সম্পর্কে কোনো তথ্য ব্যবহার করে না; শুরু থেকে স্তরে স্তরে (level by level) সব দিকে সমানভাবে ছড়ায়। branching factor b ও সমাধানের গভীরতা d হলে সবচেয়ে খারাপ ক্ষেত্রে $O(b^d)$ নোড পরীক্ষা ও মেমরিতে রাখতে হয়। সব edge-এর খরচ সমান হলে এটি সর্বনিম্ন-ধাপের পথ পায়, কিন্তু অপ্রয়োজনীয় দিকেও সমান সময় নষ্ট করে।

Heuristic (informed) search — A*: প্রতিটি নোড n-এর জন্য মূল্যায়ন ফাংশন **f(n) = g(n) + h(n)** — g(n) = শুরু থেকে n পর্যন্ত প্রকৃত খরচ, h(n) = n থেকে লক্ষ্য পর্যন্ত আনুমানিক খরচ (heuristic)। সর্বনিম্ন f-এর নোড আগে বিস্তার করে, ফলে লক্ষ্যের দিকে "অভিমুখী" হয়ে খোঁজে।

যুক্তি (কেন দ্রুত):

1. h(n) লক্ষ্য থেকে দূরে সরে যাওয়া পথগুলোর f বাড়িয়ে দেয়, তাই সেগুলো দেরিতে বা কখনোই বিস্তার হয় না — **search space ছেঁটে ফেলা (pruning)** হয়।
2. BFS-এর মতো সব দিকের সব নোড পরীক্ষা না করে লক্ষ্যের কাছাকাছি একটি সরু "করিডর" অনুসরণ করে — পরীক্ষিত নোডের সংখ্যা ও মেমরি দুটোই অনেক কমে।
3. Heuristic যত নিখুঁত (h ≈ প্রকৃত খরচ), তত কম নোড বিস্তার হয়; সর্বোত্তম ক্ষেত্রে প্রায় সরাসরি পথ।

নিখুঁত পথের নিশ্চয়তা কখন থাকে না: A* তখনই অপটিমাল পথের নিশ্চয়তা দেয় যখন h **admissible** (কখনো প্রকৃত খরচের চেয়ে বেশি অনুমান করে না) — যেমন মানচিত্রে সরলরেখার দূরত্ব। দ্রুততার জন্য অনেক সময় inadmissible বা "weighted" heuristic (f = g + w·h, w > 1) বা Greedy Best-First (শুধু h) ব্যবহার করা হয়; তখন খোঁজ আরও আক্রমণাত্মকভাবে লক্ষ্যের দিকে যায় — অনেক দ্রুত, কিন্তু মাঝে মাঝে সামান্য দীর্ঘ পথ বেছে নিতে পারে। অর্থাৎ **সময়/মেমরি বনাম অপটিমালিটির trade-off**।

উদাহরণ

ঢাকা থেকে চট্টগ্রামের রাস্তা খুঁজতে BFS ঢাকা থেকে সব দিকে (রাজশাহী, ময়মনসিংহ, খুলনা) সমানভাবে ছড়াবে। A* প্রতিটি শহর থেকে চট্টগ্রামের সরলরেখা দূরত্বকে h ধরে, ফলে উত্তর-পশ্চিমের শহরগুলোর f অনেক বড় হয় এবং কুমিল্লা-ফেনী দিকের পথগুলো আগে পরীক্ষা হয় — অনেক কম নোড ঘেঁটেই পথ পাওয়া যায়।

ভিত্তি	BFS (Blind)	A* (Heuristic)

তথ্য ব্যবহার	শুধু গঠন (graph)	লক্ষ্য সম্পর্কে অনুমান h(n)
বিস্তার ক্রম	Level by level (FIFO queue)	সর্বনিম্ন f(n) (priority queue)
পরীক্ষিত নোড	অনেক বেশি	ভালো h হলে অনেক কম
Optimality	সমান edge-cost হলে optimal	Admissible h হলে optimal; নইলে নিশ্চয়তা নেই

> **প্রশ্ন ১৪** BSCS AP 17.01.2026
>
> *Explain the logic of a "Checksum". How is it used to verify data integrity during file transfer? (10)*

Checksum হলো ডেটা থেকে গণনা করা একটি ছোট মান, যা ডেটার সাথে পাঠানো হয়; প্রাপক একই পদ্ধতিতে আবার গণনা করে মিলিয়ে দেখে ডেটা পথে বদলেছে কিনা। Internet checksum (IP/TCP/UDP header-এ ব্যবহৃত) **one's complement** যোগফলের উপর ভিত্তি করে।

প্রেরক: (১) ডেটাকে নির্দিষ্ট বিটের (যেমন 16-bit) word-এ ভাগ করে; (২) সব word one's complement পদ্ধতিতে যোগ করে (ক্যারি থাকলে আবার যোগফলে যোগ — *wrap-around*); (৩) যোগফলের complement (সব বিট উল্টে) = checksum; (৪) ডেটা + checksum পাঠায়।
প্রাপক: ডেটা + checksum সব যোগ করে (wrap-around সহ) complement নেয়; ফল **0** হলে ডেটা ঠিক আছে, নইলে ত্রুটি — ডেটা বাতিল/পুনরায় চাওয়া হয়।

উদাহরণ (4-bit word): ডেটা 7, 11, 12, 0, 6

প্রেরক: যোগফল = 7 + 11 + 12 + 0 + 6 = 36 = $(10\ 0100)_2$। 4-bit-এর বাইরে ক্যারি $10_2 = 2$ → wrap: 0100 + 0010 = 0110 (= 6)। Checksum = complement(0110) = **1001 (= 9)**। পাঠানো হয়: 7, 11, 12, 0, 6, 9।
প্রাপক: 7 + 11 + 12 + 0 + 6 + 9 = 45 = $(10\ 1101)_2$ → wrap: 1101 + 0010 = 1111 → complement = **0000** ⇒ ডেটা অক্ষত ✔
যদি পথে 11 বদলে 10 হয়ে যেত: যোগফল 44 = $(10\ 1100)_2$ → 1100 + 0010 = 1110 → complement = 0001 ≠ 0 ⇒ ত্রুটি ধরা পড়ল ✘

ফাইল ট্রান্সফারে ব্যবহার: সফটওয়্যার/ফাইল ডাউনলোডের সাইটে প্রকাশক ফাইলের **SHA-256** (আগে MD5) hash প্রকাশ করে। ডাউনলোডের পর ব্যবহারকারী নিজে hash গণনা করে মেলায় — `sha256sum file.iso` (Linux) বা `certutil -hashfile file.iso SHA256` (Windows)। মিললে ফাইলটি অক্ষত ও পরিবর্তিত হয়নি। ব্যাংক-থেকে-ব্যাংক batch ফাইল (যেমন clearing ফাইল) পাঠাতেও hash/MAC ব্যবহার হয়।

সীমাবদ্ধতা: সাধারণ checksum দুর্ঘটনাজনিত ত্রুটি ধরে, কিন্তু কিছু ত্রুটি (যেমন দুটি word-এর জায়গা বদল) ধরতে পারে না এবং ইচ্ছাকৃত পরিবর্তন রোধ করে না — আক্রমণকারী ডেটার সাথে checksum-ও বদলে দিতে পারে। শক্তিশালী ত্রুটি শনাক্তকরণে CRC, আর নিরাপত্তার জন্য cryptographic hash + digital signature/HMAC লাগে।

> **প্রশ্ন ১৫** 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 হলো AI-এর একটি শাখা যেখানে স্পষ্টভাবে প্রোগ্রাম না করে ডেটা থেকে প্যাটার্ন শিখে সিদ্ধান্ত বা ভবিষ্যদ্বাণী করা হয়। শেখার ধরন অনুযায়ী তিনটি মৌলিক paradigm:

ভিত্তি	Supervised	Unsupervised	Reinforcement
ডেটা	Labeled (input + সঠিক output)	Unlabeled (শুধু input)	লেবেল নেই; পরিবেশের সাথে মিথস্ক্রিয়া থেকে reward/penalty

লক্ষ্য	Input → output ম্যাপিং শিখে নতুন ডেটার পূর্বাভাস	লুকানো গঠন/গ্রুপ/প্যাটার্ন খোঁজা	দীর্ঘমেয়াদে মোট reward সর্বোচ্চ করে এমন policy শেখা
সমস্যার ধরন	Classification, Regression	Clustering, Association, Dimensionality reduction	Sequential decision making, control
অ্যালগরিদম	Linear/Logistic regression, Decision tree, SVM, Random forest, Neural network	K-means, Hierarchical clustering, PCA, Apriori	Q-learning, SARSA, Deep Q-Network, Policy gradient
প্রতিক্রিয়া	প্রতিটি উদাহরণে সরাসরি সঠিক উত্তর	কোনো প্রতিক্রিয়া নেই	বিলম্বিত reward সংকেত
মূল্যায়ন	Accuracy, precision, recall, RMSE	Silhouette score, অভ্যন্তরীণ মাপকাঠি	Cumulative reward
ব্যাংক উদাহরণ	ঋণ খেলাপি হবে কিনা পূর্বাভাস (credit scoring); লেনদেন জালিয়াতি কিনা শ্রেণিকরণ	গ্রাহক segmentation (মার্কেটিং); অস্বাভাবিক লেনদেন (anomaly) শনাক্ত	ডায়নামিক প্রাইসিং/ট্রেডিং কৌশল; ATM-এ নগদ পুনঃপূরণ সময়সূচি অপটিমাইজ

সাদৃশ্য: তিনটিই ডেটা থেকে মডেল শেখে এবং overfitting/ডেটার মানের সমস্যায় ভোগে। **বৈসাদৃশ্যের সারকথা:** supervised "শিক্ষকের দেওয়া উত্তর" থেকে শেখে, unsupervised নিজেই গ্রুপ খোঁজে, reinforcement "চেষ্টা-ভুল" করে পুরস্কার থেকে শেখে। এ ছাড়া *semi-supervised* (অল্প labeled + অনেক unlabeled) ও *self-supervised* (LLM প্রশিক্ষণ) পদ্ধতিও আছে।

২.৩ BSCS ২০২০ — Combined 2 Bank (Sonali & Janata) Officer (IT) লিখিত

বিষয়	তথ্য
পরীক্ষা	BSCS Recruitment 2020 — Combined 2 Bank (Sonali Bank ও Janata Bank), Officer (IT)
সময় ও নম্বর	২ ঘণ্টা, ২০০ নম্বর
কাঠামো	রচনা, অনুবাদ (B→E, E→B), ইংরেজি অনুচ্ছেদ + বিষয়ভিত্তিক প্রশ্ন (A)–(O)
কেন পড়বেন	পদ ভিন্ন হলেও এর প্রশ্নগুলো (queue, inheritance, factorial, insertion sort, SQL transaction) AP পরীক্ষার প্রশ্নের সাথে প্রায় হুবহু মিলে যায়

> **প্রশ্ন (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.*

Implementation (coding) ধাপে ডেভেলপাররা মূলত ছোট ইউনিট থেকে শুরু করে সংযুক্ত মডিউল পর্যন্ত পরীক্ষা করেন:

টেস্ট	কী ও কীভাবে
Unit Testing	প্রতিটি function/method/class আলাদাভাবে পরীক্ষা; ডেভেলপার নিজে লেখেন (JUnit, pytest); নির্ভরশীল অংশ stub/mock দিয়ে প্রতিস্থাপন। উদাহরণ: `calculateInterest(10000, 5%)` = 500 ফেরত দেয় কিনা।
Integration Testing	একক মডিউলগুলো যুক্ত করে তাদের interface ও ডেটা আদান-প্রদান পরীক্ষা। পদ্ধতি: *Big-bang* (সব একসাথে), *Top-down* (উপরের মডিউল আগে, নিচের জন্য stub), *Bottom-up* (নিচের মডিউল আগে, উপরের জন্য driver), *Sandwich* (দুটোর মিশ্রণ)।

White-box (Structural) Testing	কোডের ভেতরের লজিক দেখে test case: statement, branch, path coverage।
Black-box (Functional) Testing	শুধু input-output দেখে: equivalence partitioning, boundary value analysis (যেমন সর্বোচ্চ উত্তোলন সীমা ৫০,০০০ হলে 49,999 / 50,000 / 50,001 পরীক্ষা)।
Static Testing / Code Review	কোড না চালিয়ে peer review, walkthrough, inspection ও static analysis tool (SonarQube) দিয়ে ত্রুটি খোঁজা।
Regression & Smoke Testing	নতুন পরিবর্তনের পর পুরোনো ফিচার ঠিক আছে কিনা (regression); নতুন build মৌলিকভাবে চলে কিনা দ্রুত যাচাই (smoke)।

পরের ধাপে System testing ও User Acceptance Testing (UAT) হয়। Unit ও integration-এর ভালো কাভারেজ থাকলে পরের ধাপে ত্রুটি কম আসে এবং সংশোধনের খরচ অনেক কম হয়।

> **প্রশ্ন (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.*

Inheritance-এর মাধ্যমে একটি class অন্য class-এর বৈশিষ্ট্য ও আচরণ পায় (IS-A সম্পর্ক)। প্রকারভেদ (বিস্তারিত চিত্র ২.৫ দেখুন):

প্রকার	গঠন	উদাহরণ	ভাষার সমর্থন
Single	A → B	Account → SavingsAccount	C++, Java, Python
Multilevel	A → B → C	Person → Employee → Manager	সব
Hierarchical	A → B, A → C	Account → Savings, Current	সব
Multiple	A, B → C	Printer, Scanner → MultiFunctionDevice	C++, Python (class); Java শুধু interface দিয়ে
Hybrid	উপরের একাধিকের মিশ্রণ	Diamond: A → B, A → C, (B, C) → D	C++ (virtual inheritance দিয়ে diamond সমাধান)

Diamond problem: Multiple inheritance-এ D যদি B ও C দুজন থেকেই A-এর একই method পায়, তবে কোনটি ব্যবহার হবে তা অস্পষ্ট। C++ `virtual` base class দিয়ে, Python MRO (Method Resolution Order) দিয়ে এবং Java class-এর multiple inheritance নিষিদ্ধ করে (interface-এর default method-এ স্পষ্ট override বাধ্যতামূলক করে) এটি সমাধান করে।

```
// Java: interface দিয়ে multiple inheritance
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..."); }
}
```

প্রশ্ন (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-ভিত্তিক linear queue (আকার MAX, শুরুতে front = rear = −1):

1. Queue পূর্ণ কিনা দেখো: `rear == MAX - 1` হলে "Queue Overflow" দেখিয়ে থামো।
2. Queue খালি হলে (`front == -1`) `front = 0` করো।
3. `rear = rear + 1` — rear পয়েন্টার পরের খালি ঘরে নাও।
4. `queue[rear] = item` — নতুন উপাদান বসাও।
5. সফলতা ফেরত দাও (শেষ)।

Circular queue-তে: পূর্ণতার শর্ত `(rear + 1) % MAX == front` এবং ধাপ ৩-এ `rear = (rear + 1) % MAX` — এতে dequeue-এর পর সামনের খালি ঘর পুনর্ব্যবহার হয়।

```
#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)**। উদাহরণ: খালি queue-তে 10, 20, 30 enqueue করলে front = 0, rear = 2, queue = [10, 20, 30, _, _]।

প্রশ্ন (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.

Algorithm-এর জটিলতা মূলত দুটি সম্পদ দিয়ে মাপা হয় — **সময় (Time)** ও **মেমরি (Space)**; উভয়ই input size n-এর ফাংশন হিসেবে প্রকাশ করা হয়।

1. **Time complexity:** algorithm সম্পন্ন করতে মৌলিক অপারেশনের (তুলনা, যোগ, assignment) সংখ্যা। যেমন linear search-এ সর্বোচ্চ n তুলনা ⇒ O(n)। প্রকৃত সেকেন্ডে না মেপে অপারেশন গোনা হয়, কারণ সেকেন্ড মেশিনভেদে বদলায়।
2. **Space complexity:** মোট মেমরি = *fixed part* (কোড, ধ্রুবক, সাধারণ ভেরিয়েবল — n-এর উপর নির্ভরশীল নয়) + *variable part* (dynamic array, recursion stack — n-এর উপর নির্ভরশীল)। যেমন recursive factorial-এ stack গভীরতা n ⇒ O(n) space।
3. **Input size (n):** ডেটা যত বড়, তত বেশি কাজ — বৃদ্ধির হার (growth rate) গুরুত্বপূর্ণ: O(1) < O(log n) < O(n) < O(n log n) < $O(n^2)$ < $O(2^n)$।
4. **Input-এর প্রকৃতি/বিন্যাস:** একই আকারের ভিন্ন input-এ ভিন্ন সময় — তাই *Best, Average, Worst case* বিশ্লেষণ। যেমন insertion sort sorted ডেটায় O(n), উল্টো ক্রমে $O(n^2)$।

5. **ডেটা স্ট্রাকচারের পছন্দ:** hash table-এ খোঁজ গড়ে O(1), linked list-এ O(n)।
6. **বাহ্যিক বিষয় (asymptotic বিশ্লেষণে উপেক্ষিত):** প্রসেসরের গতি, কম্পাইলার অপটিমাইজেশন, প্রোগ্রামিং ভাষা, অপারেটিং সিস্টেম, মেমরি/ক্যাশ — এগুলো প্রকৃত চলার সময় বদলায় কিন্তু growth rate নয়।

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

> **প্রশ্ন (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	কাজ
COMMIT	Transaction-এর সব পরিবর্তন ডাটাবেসে স্থায়ী করে এবং transaction শেষ করে; এরপর আর ফেরানো যায় না (Durability)। অন্য ব্যবহারকারীরা পরিবর্তন দেখতে পান।
ROLLBACK	Transaction-এর সব (অ-commit করা) পরিবর্তন বাতিল করে ডাটাবেসকে transaction শুরুর আগের অবস্থায় ফেরায় এবং transaction শেষ করে (Atomicity)।
SAVEPOINT name	Transaction-এর ভেতরে একটি চিহ্নিত বিন্দু তৈরি করে।
ROLLBACK TO SAVEPOINT name	শুধু ওই savepoint-এর পরের পরিবর্তন বাতিল করে; transaction চলতে থাকে (শেষে COMMIT/ROLLBACK লাগবে)।

এ ছাড়া `SET TRANSACTION` দিয়ে isolation level বা read-only বৈশিষ্ট্য নির্ধারণ করা যায়। অনেক ডাটাবেসে DDL কমান্ড (CREATE, ALTER, DROP) চালালে implicit commit হয়, এবং সেশন অস্বাভাবিকভাবে বন্ধ হলে স্বয়ংক্রিয় rollback হয়। এগুলোকে একত্রে **TCL (Transaction Control Language)** বলে।

```
-- ১০,০০০ টাকা হিসাব A থেকে B-তে স্থানান্তর
START TRANSACTION;                -- (Oracle-এ স্বয়ংক্রিয়ভাবে শুরু হয়)
UPDATE account SET balance = balance - 10000 WHERE acc_no = 'A';
SAVEPOINT after_debit;
UPDATE account SET balance = balance + 10000 WHERE acc_no = 'B';
-- কোনো ত্রুটি হলে:  ROLLBACK;   (দুটি পরিবর্তনই বাতিল)
COMMIT;                           -- দুটিই স্থায়ী
```

> **প্রশ্ন (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}")
```

নমুনা রান

Input: wage = 200, regular = 40, overtime = 5 → 200 × 40 + 5 × 1.5 × 200 = 8000 + 1500 = **9500.00**
ভালো উত্তরের জন্য: ঋণাত্মক ইনপুট যাচাই (`if (wage < 0 || regular < 0 || overtime < 0)` → ত্রুটি বার্তা) যোগ করুন।

প্রশ্ন (G) BSCS Officer IT 2020

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

সংজ্ঞা: n! = n × (n − 1)!, এবং 0! = 1 (base case)। Recursion-এর দুটি অংশ — *base case* (যেখানে থামে) ও *recursive case* (ছোট সমস্যায় নিজেকে ডাকা)।

Algorithm FACT(n):
ধাপ ১: যদি n < 0 হয়, "Invalid" ফেরত দাও।
ধাপ ২: যদি n == 0 বা n == 1 হয়, 1 ফেরত দাও (base case)।
ধাপ ৩: অন্যথায় n × FACT(n − 1) ফেরত দাও (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
```

ট্রেস: fact(4)

fact(4) = 4 × fact(3) → fact(3) = 3 × fact(2) → fact(2) = 2 × fact(1) → fact(1) = 1 (base)
ফেরার পথে: fact(2) = 2 × 1 = 2 → fact(3) = 3 × 2 = 6 → fact(4) = 4 × 6 = **24**
Time O(n), Space O(n) (call stack-এ n টি frame)। খুব বড় n-এ stack overflow হতে পারে — তখন iterative সংস্করণ ভালো।

প্রশ্ন (H) BSCS Officer IT 2020

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

(ক) Scheduler-এর প্রকারভেদ (কখন সিদ্ধান্ত নেয়):

Scheduler	কাজ	ঘনত্ব
Long-term (Job) Scheduler	Job pool থেকে কোন প্রোগ্রাম মেমরিতে এনে ready queue-তে দেওয়া হবে তা ঠিক করে; *degree of multiprogramming* নিয়ন্ত্রণ করে ও CPU-bound/I/O-bound মিশ্রণ রাখে	কম (মিনিট)
Short-term (CPU) Scheduler	Ready queue থেকে কোন process পরবর্তীতে CPU পাবে তা নির্বাচন করে; dispatcher context switch করে	খুব ঘন ঘন (মিলিসেকেন্ড)
Medium-term Scheduler	মেমরি চাপ কমাতে process সাময়িকভাবে ডিস্কে সরায় (swap out) ও পরে ফেরায় (swap in)	মাঝারি

(খ) Preemptive বনাম Non-preemptive: Non-preemptive-এ একবার CPU পেলে process শেষ বা ব্লক না হওয়া পর্যন্ত চলে; preemptive-এ OS মাঝপথে CPU কেড়ে নিতে পারে (time quantum শেষ বা উচ্চ অগ্রাধিকারের process এলে)।

(গ) CPU scheduling অ্যালগরিদম:

অ্যালগরিদম	নিয়ম	বৈশিষ্ট্য
FCFS	যে আগে আসে সে আগে পায়	সহজ; convoy effect
SJF / SRTF	সবচেয়ে ছোট burst আগে (SRTF হলো preemptive সংস্করণ)	সর্বনিম্ন গড় waiting time; বড় job-এর starvation
Priority	সর্বোচ্চ অগ্রাধিকার আগে	Starvation → *aging* দিয়ে সমাধান
Round Robin	প্রত্যেকে নির্দিষ্ট time quantum পায়, তারপর queue-র শেষে	Time-sharing-এ ন্যায্য, ভালো response time
Multilevel (Feedback) Queue	একাধিক queue, ভিন্ন অগ্রাধিকার ও অ্যালগরিদম; feedback-এ process queue বদলায়	নমনীয়; আধুনিক OS-এর ভিত্তি

লক্ষ্য: CPU utilization ও throughput সর্বোচ্চ; turnaround, waiting ও response time সর্বনিম্ন।

প্রশ্ন (I) BSCS Officer IT 2020

Make a list of banking software used in Bangladesh. List the essential features for successful banking software and apps.

বাংলাদেশে ব্যবহৃত ব্যাংকিং সফটওয়্যার (ধরন অনুযায়ী):

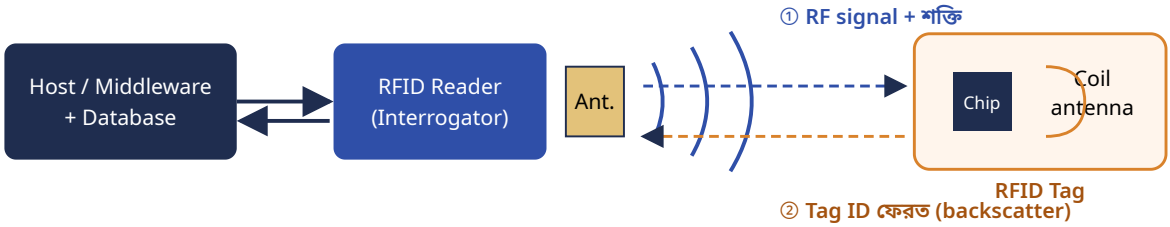
ধরন	উদাহরণ
আন্তর্জাতিক Core Banking Solution (CBS)	Temenos T24 (Transact), Oracle FLEXCUBE, Infosys Finacle
দেশীয় CBS	Bank Ultimus (Leads Corporation), Abacus (Millennium Information Solution) এবং বিভিন্ন ব্যাংকের নিজস্ব (in-house) CBS
পেমেন্ট ও আন্তঃব্যাংক	SWIFT (আন্তর্জাতিক বার্তা), বাংলাদেশ ব্যাংকের BACH (BACPS + BEFTN), RTGS, NPSB-এর সাথে সংযোগকারী সিস্টেম
ডিজিটাল চ্যানেল	Internet banking, মোবাইল অ্যাপ, ATM/POS switch, agent banking ও card management সিস্টেম
সহায়ক	AML/CFT screening, Loan origination, Treasury, MIS/Reporting সফটওয়্যার

সফল ব্যাংকিং সফটওয়্যার ও অ্যাপের অপরিহার্য বৈশিষ্ট্য:

1. **নিরাপত্তা:** MFA/OTP, end-to-end encryption (TLS), role-based access, maker-checker, audit trail, ফ্রড শনাক্তকরণ।
2. **Real-time ও কেন্দ্রীভূত লেনদেন:** যেকোনো শাখা/চ্যানেল থেকে তাৎক্ষণিক পোস্টিং (anywhere banking)।
3. **উচ্চ প্রাপ্যতা ও নির্ভরযোগ্যতা:** ২৪/৭ সেবা, DR সাইট, ব্যাকআপ, ACID-সম্মত লেনদেন।
4. **Scalability ও পারফরম্যান্স:** লাখো গ্রাহক ও উৎসবের সময় উচ্চ লেনদেন সামলানো।
5. **Integration/Interoperability:** API ও ISO 8583/ISO 20022 ভিত্তিক সংযোগ — NPSB, BEFTN, RTGS, SWIFT, MFS।
6. **নিয়ন্ত্রক কমপ্লায়েন্স:** বাংলাদেশ ব্যাংকের রিপোর্টিং, KYC/e-KYC, AML স্ক্রিনিং।
7. **ব্যবহারবান্ধব UI/UX:** সহজ নেভিগেশন, বাংলা ভাষা সমর্থন, accessibility।
8. **বহুমুখী সেবা:** হিসাব খোলা, ফান্ড ট্রান্সফার, বিল পেমেন্ট, QR পেমেন্ট, স্টেটমেন্ট, ঋণ ও কার্ড ব্যবস্থাপনা।
9. **MIS ও Analytics:** ব্যবস্থাপনার জন্য রিপোর্ট ও ড্যাশবোর্ড।
10. **রক্ষণাবেক্ষণযোগ্যতা:** modular আর্কিটেকচার, নিয়মিত আপডেট ও vendor সাপোর্ট।

> **প্রশ্ন (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) রেডিও তরঙ্গ ব্যবহার করে কোনো বস্তু/ব্যক্তিকে স্বয়ংক্রিয়ভাবে ও স্পর্শ ছাড়াই শনাক্ত করার প্রযুক্তি; বারকোডের মতো line-of-sight লাগে না।



চিত্র ২.১২: RFID সিস্টেমের মূল অংশ ও কার্যপ্রণালী

উপাদান: (১) **Tag (transponder)** — মাইক্রোচিপ (unique ID ও ডেটা সংরক্ষণ) + অ্যান্টেনা; (২) **Reader (interrogator)** — অ্যান্টেনার মাধ্যমে রেডিও সংকেত পাঠায় ও গ্রহণ করে; (৩) **Host/Middleware ও Database** — রিডারের পড়া ডেটা প্রক্রিয়া করে ব্যবসায়িক সিদ্ধান্ত নেয়।

কার্যপ্রণালী:

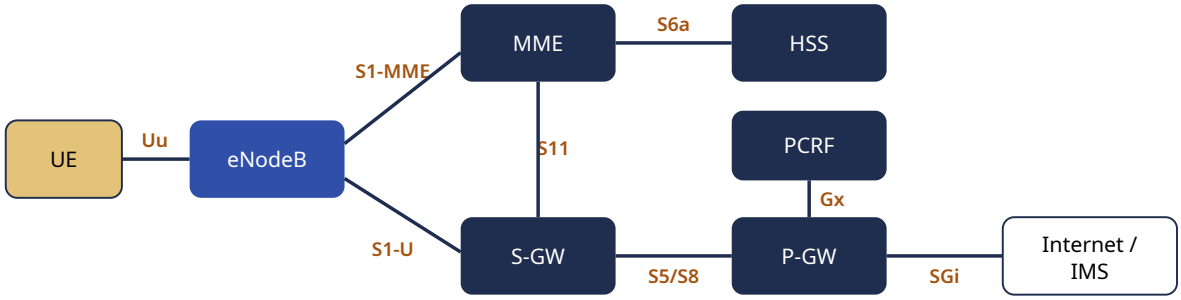
1. Reader অ্যান্টেনা দিয়ে নির্দিষ্ট ফ্রিকোয়েন্সির রেডিও তরঙ্গ ছড়ায় (interrogation zone তৈরি হয়)।
2. **Passive tag** নিজস্ব ব্যাটারি ছাড়াই অ্যান্টেনায় আবিষ্ট (induced) শক্তি দিয়ে চিপ চালু করে; **active tag** নিজস্ব ব্যাটারি দিয়ে চলে (দূরত্ব বেশি)।
3. Tag তার unique ID/ডেটা দিয়ে সংকেতকে modulate করে ফেরত পাঠায় (passive-এ *backscatter*)।
4. Reader সংকেত decode করে host-এ পাঠায়; host ডাটাবেস মিলিয়ে কাজ করে (যেমন টোল কেটে নেওয়া, ইনভেন্টরি আপডেট)।

Band	ফ্রিকোয়েন্সি	দূরত্ব	ব্যবহার
LF	125–134 kHz	~10 cm	প্রাণী শনাক্তকরণ, access card
HF	13.56 MHz	~10 cm–1 m	স্মার্ট কার্ড, লাইব্রেরি, NFC
UHF	860–960 MHz	কয়েক মিটার	সাপ্লাই চেইন, ইলেকট্রনিক টোল

প্রয়োগ: সাপ্লাই চেইনে প্যালেট/পণ্য স্বয়ংক্রিয় ট্র্যাকিং; টোল প্লাজায় গাড়ির কাচে লাগানো tag থেকে না থেমে টোল আদায়; ব্যাংকে কর্মী access control ও ভল্ট/সম্পদ ট্র্যাকিং; contactless কার্ড।

> **প্রশ্ন (K)** BSCS Officer IT 2020
>
> *Make a list of LTE network elements.*

LTE (4G) নেটওয়ার্ককে বলা হয় **EPS (Evolved Packet System)** = **E-UTRAN** (রেডিও অ্যাক্সেস) + **EPC (Evolved Packet Core)**। এটি সম্পূর্ণ IP/packet-switched; ভয়েস যায় VoLTE (IMS) দিয়ে।



চিত্র ২.১৩: LTE/EPS-এর উপাদান ও ইন্টারফেস

উপাদান	অংশ	কাজ
UE (User Equipment)	—	মোবাইল ফোন/ডিভাইস; USIM-এ গ্রাহক পরিচয় (IMSI) ও key
eNodeB (Evolved Node B)	E-UTRAN	বেস স্টেশন: রেডিও রিসোর্স ব্যবস্থাপনা, scheduling, handover; eNB-গুলো X2 ইন্টারফেসে যুক্ত (3G-র RNC-র কাজ এতেই অন্তর্ভুক্ত)
MME (Mobility Management Entity)	EPC (control plane)	Attach, authentication (HSS-এর সহায়তায়), mobility/tracking area, paging, bearer setup সংকেত

S-GW (Serving Gateway)	EPC (user plane)	ব্যবহারকারীর ডেটা প্যাকেট route ও forward; eNB-গুলোর মধ্যে handover-এ mobility anchor
P-GW (PDN Gateway)	EPC (user plane)	বাইরের নেটওয়ার্কে (Internet/IMS) সংযোগ, UE-কে IP ঠিকানা বরাদ্দ, policy enforcement, charging
HSS (Home Subscriber Server)	EPC	গ্রাহকের প্রোফাইল ও authentication তথ্যের কেন্দ্রীয় ডাটাবেস (HLR-এর উত্তরসূরি)
PCRF (Policy and Charging Rules Function)	EPC	QoS ও চার্জিং নীতি নির্ধারণ করে P-GW-কে জানায়

প্রশ্ন (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.

শ্রেণি	কাজ	কমান্ড	উদাহরণ
DDL	ডাটাবেসের গঠন (schema) সংজ্ঞায়িত/পরিবর্তন; সাধারণত 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	টেবিলের ডেটা পড়া ও পরিবর্তন	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	ব্যবহারকারীর অনুমতি নিয়ন্ত্রণ	GRANT, REVOKE	`GRANT SELECT, INSERT ON customer TO teller1;` `REVOKE INSERT ON customer FROM teller1;`
TCL (অতিরিক্ত)	Transaction নিয়ন্ত্রণ	COMMIT, ROLLBACK, SAVEPOINT	`COMMIT;` `ROLLBACK;`

টীকা: কিছু বইয়ে SELECT-কে আলাদা DQL (Data Query Language) হিসেবে দেখানো হয়। TRUNCATE DDL (সব সারি মুছে, rollback সাধারণত হয় না), কিন্তু DELETE DML (শর্তসহ, rollback করা যায়)।

প্রশ্ন (M) BSCS Officer IT 2020

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

(i) $(10010.101)_2$ → দশমিক

পূর্ণ অংশ: $1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 = 16 + 0 + 0 + 2 + 0 = 18$

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

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

(ii) $(543)_{10}$ → হেক্সাডেসিমাল (বারবার 16 দিয়ে ভাগ)

ভাগ	ভাগফল	ভাগশেষ	Hex অঙ্ক
543 ÷ 16	33	15	F
33 ÷ 16	2	1	1
2 ÷ 16	0	2	2

ভাগশেষগুলো নিচ থেকে উপরে পড়ে: **$(543)_{10} = (21F)_{16}$**

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

প্রশ্ন (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.

লজিক: Array-কে দুই অংশে ভাবা হয় — বামের sorted অংশ ও ডানের unsorted অংশ। প্রতি ধাপে unsorted অংশের প্রথম উপাদান (key) নিয়ে sorted অংশের বড় উপাদানগুলো এক ঘর ডানে সরিয়ে key-কে সঠিক স্থানে বসানো হয় — তাস হাতে সাজানোর মতো।

```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];   /* ডানে সরাও */
            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 টেবিল: [12, 11, 13, 5, 6]

i	key	কাজ	Array (পরে)
1	11	12 ডানে সরে, 11 index 0-এ	11 12 | 13 5 6
2	13	13 > 12 — নড়াচড়া নেই	11 12 13 | 5 6
3	5	13, 12, 11 ডানে সরে, 5 index 0-এ	5 11 12 13 | 6
4	6	13, 12, 11 ডানে সরে, 6 index 1-এ	**5 6 11 12 13**

Complexity: Best O(n) (আগে থেকে sorted), Average/Worst O(n^2); Space O(1); stable ও in-place — ছোট বা প্রায়-সাজানো ডেটায় খুব কার্যকর।

প্রশ্ন (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?

IP স্তর best-effort — প্যাকেট হারাতে, দেরিতে আসতে, ক্রম বদলাতে বা দুবার আসতে পারে। TCP নিচের কৌশলগুলো দিয়ে নির্ভরযোগ্য byte-stream সেবা দেয়:

1. **Connection management:** ডেটা পাঠানোর আগে 3-way handshake (SYN → SYN-ACK → ACK) দিয়ে উভয় পক্ষ প্রস্তুত ও initial sequence number বিনিময়; শেষে 4-way (FIN/ACK) দিয়ে সুশৃঙ্খল সমাপ্তি।
2. **Sequence number:** প্রতিটি byte-এর ক্রমিক নম্বর থাকে, ফলে প্রাপক ভুল ক্রমে আসা segment সঠিক ক্রমে সাজায় এবং **duplicate segment শনাক্ত করে বাদ দেয়**।
3. **Acknowledgement (ACK):** প্রাপক cumulative ACK দিয়ে জানায় পরবর্তী কোন byte আশা করছে।
4. **Retransmission:** নির্দিষ্ট সময়ে (RTO — RTT থেকে গণনা) ACK না এলে segment আবার পাঠায়; ৩টি duplicate ACK পেলে timeout-এর অপেক্ষা না করে *fast retransmit*।
5. **Error detection — Checksum:** প্রতিটি segment-এ (header + data + pseudo-header) checksum; নষ্ট segment বাতিল হয় এবং পরে পুনরায় পাঠানো হয়।
6. **Flow control:** প্রাপক receive window (rwnd) জানিয়ে দেয় কতটা ডেটা গ্রহণ করতে পারবে — sliding window প্রেরককে ধীর প্রাপককে ডুবিয়ে দেওয়া থেকে বিরত রাখে।
7. **Congestion control:** নেটওয়ার্কের যানজট বুঝে প্রেরণের হার নিয়ন্ত্রণ — slow start, congestion avoidance (AIMD), fast recovery — ফলে প্যাকেট হারানো কমে।
8. **In-order, byte-stream delivery:** অ্যাপ্লিকেশন শুধু ক্রমানুসারে, ফাঁকহীন ডেটা পায়।

উত্তর লেখার টিপস

"Functions" প্রশ্নে ৬–৮টি বুলেট + একটি ছোট 3-way handshake চিত্র (Client → SYN → Server; Server → SYN+ACK → Client; Client → ACK → Server) দিলে পূর্ণ নম্বর পাওয়ার সম্ভাবনা বাড়ে।

২.৪ Paper F — BSCS Assistant Programmer (Job ID-25106), ৩১.১০.২০২৬ — সম্পূর্ণ প্রশ্নপত্র

বিষয়	তথ্য

পরীক্ষা	Recruitment Test 2026 (Written Part) — Assistant Programmer / Senior Officer (Assistant Programmer), Job ID-25106 — **যে পরীক্ষার জন্য এই বই লেখা**
তারিখ	৩১/১০/২০২৬ (বিজ্ঞপ্তি নং ৩৪০/২০২৬)
লিখিত	২ ঘণ্টা, ২০০ নম্বর: Subject Related প্রশ্ন ১–১৫ (১৫ × ১০ = ১৫০) + Mathematics প্রশ্ন ১৬–১৯ (৪ × ৫ = ২০) + English Proficiency প্রশ্ন ২০–২১ (১০ + ২০ = ৩০)
নতুন যা এসেছে	আধুনিক AI নিয়ে চারটি প্রশ্ন (NLP, Generative AI/LLM, MCP-সহ agentic AI, RAG); দুটি গাণিতিক হিসাবের প্রশ্ন (capacity planning, cache AMAT); একটি BST trace; UML-সহ একটি OOP design; ২০ নম্বরের একটি English essay

অংশ ২.১–২.৩-এর মতো নয়, এই পেপারটি এখানে **সম্পূর্ণভাবে** সমাধান করা হলো — গণিত ও English অংশসহ — কারণ এটি এই পদের সবচেয়ে সাম্প্রতিক পেপার এবং পরের পরীক্ষার সবচেয়ে ভালো দিকনির্দেশক। প্রতিটি technical উত্তর এমন দৈর্ঘ্য ও কাঠামোয় লেখা, যা ৭–৮ মিনিটে লিখে পূর্ণ নম্বর পাওয়া যায়।

> **প্রশ্ন ১** BSCS AP 31.10.2026
>
> *Explain the OSI reference model and the functions of its seven layers. (10)*

OSI (Open Systems Interconnection) reference model ১৯৮৪ সালে ISO প্রকাশ করে; এটি নেটওয়ার্ক যোগাযোগকে **সাতটি layer**-এ ভাগ করে। প্রতিটি layer একটি নির্দিষ্ট কাজ করে, নিচের layer-এর সেবা ব্যবহার করে এবং উপরের layer-কে সেবা দেয় — ফলে ভিন্ন ভিন্ন vendor-এর hardware ও software একসাথে কাজ করতে পারে। ডেটা পাঠানোর সময় প্রতিটি layer নিজস্ব header যোগ করে (**encapsulation**); প্রাপক উল্টো ক্রমে সেগুলো সরিয়ে নেয় (decapsulation)।

#	Layer	প্রধান কাজ	Data unit	উদাহরণ
7	**Application**	ব্যবহারকারীর অ্যাপ্লিকেশন ও নেটওয়ার্কের মধ্যে interface: ই-মেইল, file transfer, ওয়েব অ্যাক্সেস, name resolution	Data	HTTP, FTP, SMTP, DNS
6	**Presentation**	ডেটা ফরম্যাটের রূপান্তর (character code), encryption/decryption, compression	Data	TLS, JPEG, ASCII/Unicode
5	**Session**	Session স্থাপন, রক্ষণাবেক্ষণ ও সমাপ্তি; dialog control (half/full duplex); recovery-র জন্য checkpoint	Data	RPC, NetBIOS
4	**Transport**	Port number ব্যবহার করে end-to-end (process-to-process) delivery; segmentation ও reassembly; flow control; error control ও retransmission	Segment	TCP, UDP
3	**Network**	Logical (IP) addressing; routing — বিভিন্ন নেটওয়ার্কের মধ্যে পথ নির্বাচন; 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**	কাঁচা bit-কে বৈদ্যুতিক, আলোক বা রেডিও সিগন্যাল হিসেবে পাঠায়; cable, connector, voltage, data rate, bit synchronisation নির্ধারণ করে	Bit	UTP, fibre, hub, repeater

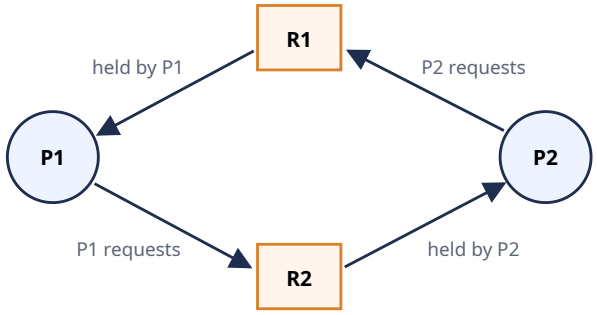
পূর্ণ নম্বর পেতে

উপর থেকে নিচে mnemonic "**A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing", encapsulation নিয়ে এক লাইন, এবং TCP/IP নিয়ে এক লাইন যোগ করুন: TCP/IP মডেল OSI-র ৫–৭ মিলিয়ে একটি Application layer এবং ১–২ মিলিয়ে একটি Network Access layer বানায়। উদাহরণ: গ্রাহক ইন্টারনেট ব্যাংকিং খুললে HTTP (L7) ডেটা TLS দিয়ে encrypt হয় (L6), TCP segment-এ port 443-এ যায় (L4), IP দিয়ে route হয় (L3), Ethernet দিয়ে frame হয় (L2) এবং fibre দিয়ে সিগন্যাল হিসেবে পাঠানো হয় (L1)।

প্রশ্ন ২ BSCS AP 31.10.2026

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

Deadlock হলো এমন একটি অবস্থা যেখানে কয়েকটি process চিরকালের জন্য block হয়ে থাকে, কারণ প্রতিটি process একটি resource ধরে রেখে একই দলের অন্য process-এর ধরে রাখা resource-এর জন্য অপেক্ষা করে। কেউই এগোতে পারে না, resource ছাড়তে পারে না, বা জাগ্রতও হতে পারে না।



চিত্র ২.১৪: Cycle-যুক্ত resource-allocation graph — P1 ও P2 deadlock-এ আটকে আছে

চারটি প্রয়োজনীয় শর্ত (Coffman conditions) — চারটি শর্ত একই সময়ে সত্য হলেই কেবল deadlock ঘটতে পারে:

1. **Mutual exclusion:** অন্তত একটি resource ভাগ করে ব্যবহার করা যায় না — একসময়ে কেবল একটি process সেটি ব্যবহার করতে পারে।
2. **Hold and wait:** একটি process অন্তত একটি resource ধরে রেখে অন্য resource-এর জন্য অপেক্ষা করে।
3. **No preemption:** কোনো resource জোর করে কেড়ে নেওয়া যায় না; process স্বেচ্ছায় ছাড়লে তবেই মুক্ত হয়।
4. **Circular wait:** P1 → P2 → … → Pn → P1 — এমন একটি চক্র থাকে, যেখানে প্রত্যেকে পরেরজনের ধরে রাখা resource-এর জন্য অপেক্ষা করে।

Deadlock সামলানোর পদ্ধতি

পদ্ধতি	মূল ধারণা	কীভাবে
Prevention	নিশ্চিত করা যে অন্তত একটি শর্ত কখনোই সত্য হবে না	Hold and wait: সব resource একবারে চাওয়া। No preemption: অনুরোধ ব্যর্থ হলে হাতে থাকা resource ছেড়ে দেওয়া। Circular wait: resource-গুলোকে নম্বর দিয়ে সবসময় ঊর্ধ্বক্রমে চাওয়া। (Mutual exclusion সাধারণত দূর করা যায় না।)
Avoidance	অনুরোধ মঞ্জুর করা হয় কেবল তখনই, যখন সিস্টেম *safe state*-এ থাকে	প্রতিটি process আগেই তার সর্বোচ্চ চাহিদা জানায়; **Banker's algorithm** যাচাই করে এমন কোনো ক্রম আছে কিনা, যাতে প্রতিটি process শেষ করতে পারে

Detection and recovery	Deadlock ঘটতে দেওয়া, তারপর খুঁজে বের করে ভেঙে দেওয়া	Wait-for graph-এ cycle খোঁজা; process abort করে (সবগুলো, বা একটি একটি করে) অথবা resource preempt করে একটি victim-কে rollback করে recovery
Ignore (Ostrich)	ধরে নেওয়া যে deadlock কদাচিৎ ঘটে	Windows ও Linux-এর মতো general-purpose OS এটি ব্যবহার করে; ব্যবহারকারী বা administrator আটকে যাওয়া process restart করেন

ব্যাংকিং উদাহরণ

Transaction T1 হিসাব A lock করে তারপর B চায়; T2 হিসাব B lock করে তারপর A চায় — এটি circular wait। DBMS এটি শনাক্ত করে একটি transaction rollback করে (detection and recovery)। আরও ভালো উপায়: সবসময় হিসাব নম্বরের ঊর্ধ্বক্রমে হিসাব lock করা — এতে circular wait দূর হয় (prevention)।

প্রশ্ন ৩ BSCS AP 31.10.2026

Explain the ACID properties of database transactions. (10)

Transaction হলো কয়েকটি database operation-এর একটি দল, যা কাজের একটি logical একক গঠন করে — যেমন fund transfer। ব্যর্থতা এবং অনেক ব্যবহারকারীর একযোগে কাজের মধ্যেও ডেটা যেন সঠিক থাকে, সেজন্য DBMS চারটি বৈশিষ্ট্যের নিশ্চয়তা দেয় — **ACID**।

```
-- 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
```

বৈশিষ্ট্য	অর্থ	Transfer-এর ক্ষেত্রে	DBMS কীভাবে নিশ্চিত করে
Atomicity	হয় সব, নয় কিছুই না — প্রতিটি operation সম্পন্ন হবে, অথবা একটিও হবে না	A থেকে debit হওয়ার পর server crash করলে debit-টি বাতিল (undo) হয়; টাকা কখনো হারায় না	Undo log, rollback
Consistency	Transaction database-কে একটি বৈধ অবস্থা থেকে আরেকটি বৈধ অবস্থায় নেয়; সব নিয়ম ও constraint বজায় থাকে	A + B-এর মোট পরিমাণ অপরিবর্তিত থাকে; ব্যালান্স কখনো অনুমোদিত ন্যূনতম সীমার নিচে নামে না	Constraint (CHECK, foreign key), trigger, application rule
Isolation	একযোগে চলা transaction-গুলো একে অপরের uncommitted কাজ দেখে না; ফলাফল এমন হয় যেন সেগুলো একটির পর একটি চলেছে	Transfer চলাকালে ব্যালান্স জানতে চাইলে কখনো দেখা যাবে না যে A থেকে debit হয়েছে কিন্তু B-তে এখনো credit হয়নি	Locking (2PL), MVCC; isolation level Read Uncommitted → Serializable
Durability	একবার commit হলে পরিবর্তন বিদ্যুৎ বিভ্রাট বা crash-এর পরও টিকে থাকে	"Transfer successful" SMS আসার পর transfer-টি স্থায়ী	Write-ahead log (redo log), disk-এ flush, replication

Isolation না থাকলে কিছু অসংগতি (anomaly) ঘটে: **dirty read** (uncommitted ডেটা পড়া), **non-repeatable read** (একই row ভিন্ন মান দেয়) এবং **phantom read** (পুনরায় চালানো query-তে নতুন row দেখা দেয়)। Core banking system ACID-এর ওপর নির্ভর করে; অনেক NoSQL system scale-এর জন্য এটি শিথিল করে তার বদলে BASE (Basically Available, Soft state, Eventually consistent) দেয়।

প্রশ্ন ৪ BSCS AP 31.10.2026

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

Natural Language Processing (NLP) হলো artificial intelligence-এর একটি শাখা, যা কম্পিউটারকে মানুষের ভাষা — লেখা ও কথা — বুঝতে, ব্যাখ্যা করতে ও তৈরি করতে সক্ষম করে। এটি ভাষাবিজ্ঞানকে machine learning-এর সাথে যুক্ত করে। এর দুটি দিক: **Natural Language Understanding** (লেখা থেকে অর্থ বের করা) এবং **Natural Language Generation** (লেখা তৈরি করা)।

প্রধান কৌশল (একটি সাধারণ pipeline)

1. **Text pre-processing:** *tokenization* (লেখাকে শব্দ বা sub-word-এ ভাগ করা), lower-casing, *stop-word removal* (the, is, a), *stemming* (running → run, কেটে ছেঁটে) এবং *lemmatization* (better → good, অভিধান ব্যবহার করে)।
2. **Syntactic analysis:** *part-of-speech tagging* (noun, verb ...) এবং *parsing* (বাক্যের ব্যাকরণগত গঠন তৈরি করা)।
3. **Semantic analysis:** *named entity recognition* (ব্যক্তি, প্রতিষ্ঠান, টাকার অংক, তারিখ শনাক্ত করা), word-sense disambiguation, coreference resolution।
4. **Text representation:** Bag-of-Words ও **TF-IDF** (দুর্লভতা অনুযায়ী ওজন দেওয়া শব্দ-গণনা); Word2Vec ও GloVe-এর মতো **word embedding**, যা একই অর্থের শব্দগুলোকে vector হিসেবে কাছাকাছি রাখে।
5. **Machine learning ও deep learning model:** classification-এর জন্য Naive Bayes ও SVM; sequence-এর জন্য RNN/LSTM; বর্তমানে মূলত self-attention-ভিত্তিক **Transformer** (বোঝার জন্য BERT, তৈরির জন্য GPT-ধাঁচের model)।

প্রয়োগ

- **Chatbot ও virtual assistant** — ব্যালান্স জানা, কার্ড block করার মতো কাজে ২৪/৭ গ্রাহকসেবা।
- **Machine translation** — বাংলা ↔ English (Google Translate)।
- **Sentiment analysis** — review ও সামাজিক মাধ্যম থেকে গ্রাহকের মতামত পরিমাপ।
- ই-মেইল ও SMS-এ **spam ও phishing শনাক্তকরণ**।
- **Speech recognition ও text-to-speech** — voice banking, IVR।
- **Text summarisation, information extraction** — KYC ফর্ম, চুক্তিপত্র, ঋণের নথি থেকে ডেটা বের করা।
- **Search engine ও question answering**।

চ্যালেঞ্জ: অস্পষ্টতা (ambiguity), ব্যঙ্গ (sarcasm), প্রসঙ্গ (context), এবং বাংলার মতো low-resource ভাষার জন্য সীমিত ডেটা।

প্রশ্ন ৫ 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 হলো এমন AI, যা শুধু শ্রেণিবিভাগ বা label পূর্বাভাস না করে training ডেটার প্যাটার্ন শিখে **নতুন কনটেন্ট** — লেখা, ছবি, অডিও, ভিডিও বা code — তৈরি করে। উদাহরণ: ChatGPT, Claude ও Gemini (লেখা), DALL·E ও Midjourney (ছবি)। **Large Language Model (LLM)** হলো বিলিয়ন বিলিয়ন parameter-যুক্ত একটি generative model, যা ভাষা বোঝা ও তৈরি করার জন্য বিপুল পরিমাণ লেখার ওপর train করা হয়।

LLM কীভাবে কাজ করে

1. **Tokenization:** input লেখাকে token-এ (শব্দ বা শব্দাংশ) ভাগ করা হয়, প্রতিটিকে একটি সংখ্যায় ম্যাপ করা হয়।

2. **Embedding:** প্রতিটি token একটি vector-এ পরিণত হয়; positional তথ্য যোগ করা হয়, যাতে model শব্দের ক্রম বোঝে।
3. **Transformer layer:** মূল অংশ হলো **self-attention** mechanism — প্রতিটি token Query, Key ও Value vector গণনা করে এবং ঠিক করে অন্য প্রতিটি token-এর প্রতি কতটা "মনোযোগ" দেবে; এভাবে প্রসঙ্গ ধরা পড়ে (যেমন নদীর "bank" বনাম bank account)। এরপর feed-forward layer থাকে; এমন কয়েক ডজন layer একটির ওপর আরেকটি সাজানো থাকে।
4. **Next-token prediction:** শেষ layer vocabulary-র প্রতিটি token-এর জন্য একটি probability দেয়; একটি token বাছাই করা হয় (*temperature* দিয়ে নিয়ন্ত্রিত), input-এর সাথে জুড়ে দেওয়া হয়, এবং প্রক্রিয়াটি আবার চলে — একে বলে **autoregressive** generation।
5. **Training:** (ক) self-supervised next-token prediction দিয়ে ট্রিলিয়ন ট্রিলিয়ন token-এ *pre-training*; (খ) instruction-এর ওপর *fine-tuning*; (গ) সহায়ক ও নিরাপদ করতে মানুষের feedback দিয়ে *alignment* (RLHF)।

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

প্রয়োগ	চ্যালেঞ্জ
• গ্রাহকসেবা chatbot ও assistant • Code generation ও debugging • রিপোর্ট, সার্কুলার ও চুক্তিপত্রের সারসংক্ষেপ • অনুবাদ ও কনটেন্ট লেখা • নথির ওপর search ও question answering (RAG) • শিক্ষা, টিউটরিং, data analysis	• **Hallucination** — আত্মবিশ্বাসী কিন্তু ভুল উত্তর • Training ডেটা থেকে আসা bias • গোপনীয়তা ও গোপন ডেটা ফাঁস • নিরাপত্তা: prompt injection, phishing বা deepfake-এ অপব্যবহার • উচ্চ compute ও জ্বালানি খরচ • Knowledge cutoff, সীমিত explainability, copyright সমস্যা

তাই ব্যাংকগুলো LLM ব্যবহার করে সুরক্ষাব্যবস্থাসহ: private deployment, অনুমোদিত নথি থেকে retrieval, মানুষের পর্যালোচনা ও logging।

> **প্রশ্ন ৬** BSCS AP 31.10.2026
>
> *What are a compiler, an interpreter, and an assembler? Explain their functions and differences. (10)*

তিনটিই **language translator** — এমন system software, যা এক ভাষায় লেখা প্রোগ্রামকে CPU চালাতে পারে এমন machine code-এ রূপান্তর করে।

- **Compiler: পুরো** high-level প্রোগ্রাম (source code) চালানোর আগেই একবারে machine/object code-এ অনুবাদ করে এবং একটি executable ফাইল তৈরি করে। পুরো প্রোগ্রাম বিশ্লেষণের পর সব error একসাথে জানায়। ধাপ: lexical analysis → syntax analysis → semantic analysis → intermediate code → optimisation → code generation। উদাহরণ: GCC (C/C++), javac (Java → bytecode)।
- **Interpreter:** প্রোগ্রামকে **একবারে একটি statement** করে অনুবাদ ও execute করে, আলাদা কোনো executable তৈরি করে না। প্রথম error-এ থেমে যায়। উদাহরণ: Python, JavaScript engine, Ruby, PHP।
- **Assembler: assembly language** (`MOV`, `ADD`-এর মতো mnemonic) থেকে machine code তৈরি করে, সাধারণত একটি instruction থেকে একটি machine instruction। উদাহরণ: NASM, MASM।

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

ভিত্তি	Compiler	Interpreter	Assembler
Input	High-level language	High-level language	Assembly language
অনুবাদের একক	পুরো প্রোগ্রাম একবারে	লাইন বাই লাইন	পুরো assembly প্রোগ্রাম
আউটপুট	Object code / executable	কোনো object ফাইল নেই; সরাসরি চলে	Machine (object) code
Execution গতি	দ্রুত (আগেই অনূদিত)	ধীর (প্রতিবার চালানোর সময় অনুবাদ)	দ্রুত
Error রিপোর্ট	Compilation-এর পর সব error	প্রথম error-এ থেমে যায়	Assemble করার পর
মেমরি	বেশি (object code সংরক্ষিত থাকে)	কম	কম
Debugging	তুলনামূলক কঠিন	সহজ	সবচেয়ে কঠিন (low level)
উদাহরণ	C, C++, Go	Python, JavaScript	NASM, MASM

Hybrid: Java প্রথমে `javac` দিয়ে bytecode-এ compile হয়, তারপর JVM সেটি interpret করে এবং ঘন ঘন ব্যবহৃত অংশগুলোকে **JIT (Just-In-Time) compiler** দিয়ে native code-এ compile করে — portability ও গতি দুটোই।

> **প্রশ্ন ৭** 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) হলো client (browser/app) ও server-এর মধ্যে ওয়েব কনটেন্ট আদান-প্রদানের application-layer protocol। এটি stateless। **HTTPS** হলো **TLS**-encrypted সংযোগের ভেতর দিয়ে বহন করা HTTP।

ভিত্তি	HTTP	HTTPS
নিরাপত্তা	Plain text — পথে পড়া বা বদলানো যায়	TLS দিয়ে encrypted — confidentiality, integrity ও server authentication
Default port	80	443
Certificate	লাগে না	Certificate Authority থেকে SSL/TLS certificate লাগে
URL / browser	http:// — "Not secure"	https:// — padlock (তালা চিহ্ন)
আক্রমণ	Eavesdropping, man-in-the-middle, session hijacking	এগুলো থেকে সুরক্ষিত
ব্যবহার	কেবল public, অসংবেদনশীল কনটেন্টের জন্য	Login, payment, ইন্টারনেট ব্যাংকিং-এর জন্য বাধ্যতামূলক; SEO-র জন্যও ভালো

HTTP request-response cycle

1. **DNS lookup:** browser domain name-কে (যেমন ibanking.examplebank.com) IP address-এ রূপান্তর করে।
2. **TCP connection:** server-এর সাথে port 80/443-এ 3-way handshake (SYN, SYN-ACK, ACK)।
3. **TLS handshake** (কেবল HTTPS): server তার certificate পাঠায়, client সেটি যাচাই করে, এবং দুই পক্ষ session key-তে একমত হয়।
4. **Request:** client একটি request line (method, path, version), header এবং ঐচ্ছিক body পাঠায়।
5. **Processing:** server request-টি route করে, application logic চালায়, database-এ query করে।
6. **Response:** server একটি status line (status code), header ও body (HTML/JSON) ফেরত দেয়।

7. **Render / close:** browser পেজটি render করে; পরবর্তী request-এর জন্য connection খোলা রাখা হয় (keep-alive) অথবা বন্ধ করা হয়।

```
GET /api/accounts/1001/balance HTTP/1.1        HTTP/1.1 200 OK
Host: ibanking.examplebank.com                 Content-Type: application/json
Authorization: Bearer eyJhbGci...
Accept: application/json                       {"account":"1001","balance":52340.75}
```

Method	উদ্দেশ্য	Safe / Idempotent
GET	একটি resource পড়া/আনা	হ্যাঁ / হ্যাঁ
POST	নতুন resource তৈরি বা ডেটা জমা দেওয়া (যেমন fund transfer)	না / না
PUT	একটি resource সম্পূর্ণ প্রতিস্থাপন	না / হ্যাঁ
PATCH	Resource-এর অংশবিশেষ হালনাগাদ	না / সবসময় নয়
DELETE	একটি resource মুছে ফেলা	না / হ্যাঁ
HEAD, OPTIONS	শুধু header; কোন কোন method অনুমোদিত তা জানতে চাওয়া (CORS)	হ্যাঁ / হ্যাঁ

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

প্রশ্ন ৮ 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)

দুর্বলতা: SQL Injection (SQLi) — OWASP Top 10 (A03: Injection)। ব্যবহারকারীর input সরাসরি SQL লেখার মধ্যে জুড়ে দেওয়ায় database ডেটা আর code আলাদা করতে পারে না। আক্রমণকারী input box-এ SQL syntax লিখে query-র অর্থই বদলে দিতে পারে। এই design থেকে আরও বোঝা যায় যে password plain text-এ সংরক্ষিত — যা দ্বিতীয় একটি গুরুতর ত্রুটি।

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

Exploit — authentication এড়িয়ে যাওয়া

Input	Database আসলে যে query চালায়	ফলাফল
Username: `admin' --` Password: `anything`	`SELECT * FROM users WHERE username = 'admin' --' AND password = 'anything'`	`--` password যাচাইয়ের অংশটিকে comment বানিয়ে দেয় ⇒ **admin** হিসেবে login

Username: `' OR '1'='1` Password: `' OR '1'='1`	`... WHERE username = '' OR '1'='1' AND password = '' OR '1'='1'`	শর্তটি সবসময় সত্য ⇒ সব row ফেরত আসে ⇒ প্রথম ব্যবহারকারী (প্রায়ই administrator) হিসেবে login

আরও ক্ষতি: `UNION SELECT` দিয়ে অন্য table পড়া, ব্যালান্স বদলানো, অথবা stacked query অনুমোদিত থাকলে `'; DROP TABLE users; --`।

প্রতিরোধ

1. **Parameterised query / prepared statement** (মূল সমাধান): SQL-এর কাঠামো আগে স্থির করা হয়; input আলাদাভাবে পাঠানো হয় এবং সবসময় ডেটা হিসেবেই গণ্য হয়। Stored procedure ও ORM (Hibernate, JPA) সঠিকভাবে ব্যবহার করলে একই সুরক্ষা পাওয়া যায়।
2. **Input validation** — username-এর দৈর্ঘ্য ও অক্ষরের জন্য allow-list।
3. **Least privilege** — অ্যাপ্লিকেশনের DB account table drop করতে বা অপ্রাসঙ্গিক ডেটা পড়তে পারবে না।
4. **Generic error message** — ব্যবহারকারীকে কখনো SQL error দেখানো যাবে না।
5. **Web Application Firewall**, code review, SAST/DAST ও penetration testing; account lockout ও 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
```

Password নিরাপদে সংরক্ষণ

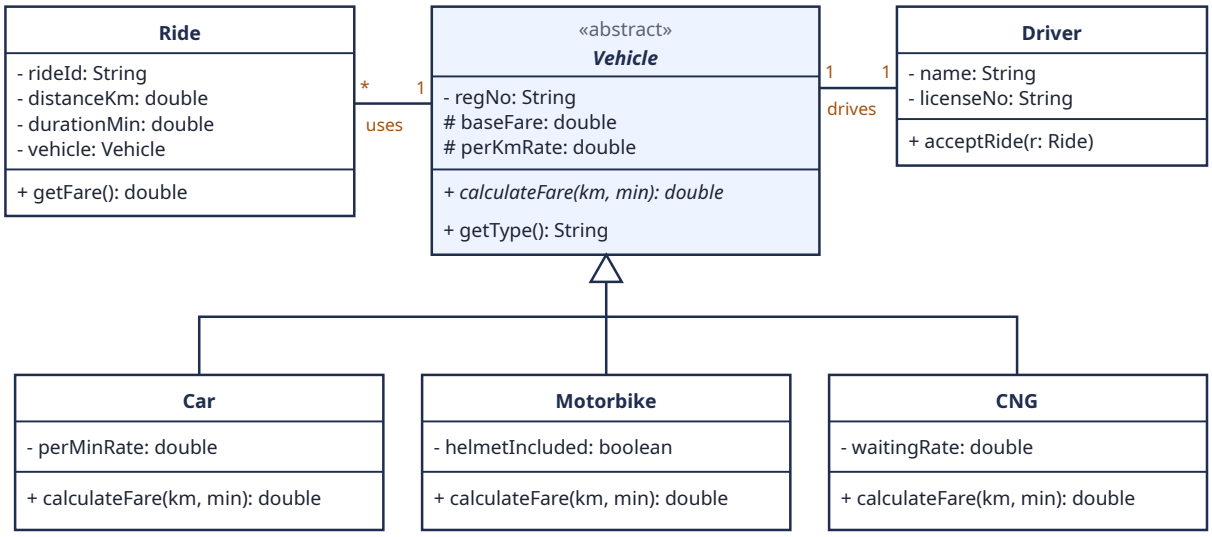
- কখনো plain text বা reversible encryption-এ রাখা যাবে না; কেবল **hash** সংরক্ষণ করতে হবে।
- ধীর, salted password-hashing algorithm ব্যবহার করুন: **Argon2id, bcrypt** বা PBKDF2 — MD5 বা SHA-1 নয়, কারণ এগুলো দ্রুত এবং সহজে ভাঙা যায়।
- প্রতিটি ব্যবহারকারীর জন্য আলাদা random **salt** rainbow table আক্রমণ ব্যর্থ করে; ঐচ্ছিক গোপন **pepper** database-এর বাইরে রাখা হয়।
- শক্তিশালী password policy, **multi-factor authentication** (OTP) এবং login ফর্মের জন্য HTTPS বাধ্যতামূলক করুন।

প্রশ্ন ৯ 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-এর মূল ধারণা: প্রতিটি যানবাহনের ভাড়া আছে, কিন্তু প্রতিটি ধরন তা ভিন্নভাবে হিসাব করে। তাই আমরা একটি abstract class `Vehicle` সংজ্ঞায়িত করি, যার একটি abstract method `calculateFare()`, এবং প্রতিটি যানবাহনের ধরন সেটি override করে। `Ride` class সঠিক ধরন না জেনেই `vehicle.calculateFare()` call করে।

OOP নীতি	কোথায় ব্যবহৃত
Abstraction	`Vehicle` abstract এবং body ছাড়াই `calculateFare()` ঘোষণা করে — এটি *কী* করতে হবে বলে, *কীভাবে* নয়
Inheritance	`Car`, `Motorbike` ও `CNG` `Vehicle`-কে extend করে এবং এর সাধারণ field-গুলো (regNo, baseFare, perKmRate) পুনর্ব্যবহার করে
Polymorphism	প্রতিটি subclass `calculateFare()` override করে; runtime-এ সঠিক version-টি চলে (dynamic binding)
Encapsulation	Field-গুলো `private` / `protected`; রেট কেবল method-এর মাধ্যমেই বদলায়
Open/Closed principle	নতুন ধরন (যেমন `Ambulance`) `Ride` না বদলেই নতুন subclass হিসেবে যোগ করা যায়



চিত্র ২.১৫: UML class diagram — প্রতিটি যানবাহন থেকে abstract Vehicle-এর দিকে generalisation (ফাঁপা ত্রিভুজ); association Ride–Vehicle (* থেকে 1) ও Driver–Vehicle (1 থেকে 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
}
```

একই যাত্রা, তিন রকম ভাড়া (8 km, 25 মিনিট; উদাহরণস্বরূপ রেট)

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

`new Ride(new CNG("DHAKA-THA-11-2345"), 8, 25).getFare()` → 300। পরে surge pricing বা ছাড় যোগ করতে **Strategy pattern** (আলাদা একটি `FareStrategy` interface) ব্যবহার করা যায়।

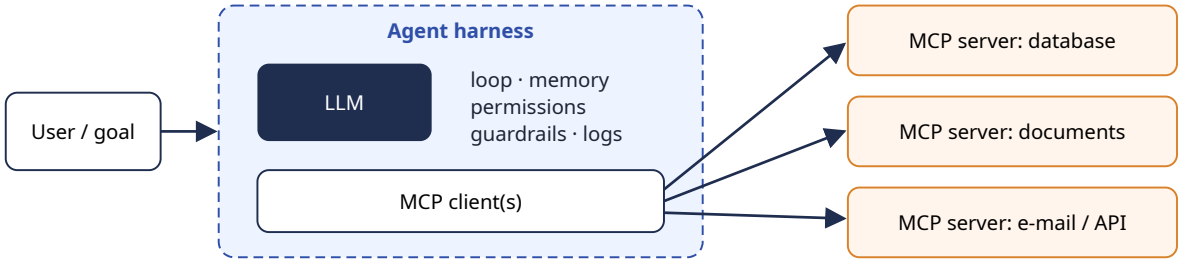
প্রশ্ন ১০ BSCS AP 31.10.2026

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)

Agentic AI system হলো সাধারণত LLM-এর ওপর তৈরি এমন একটি AI system, যা একটি প্রম্পটে একটি উত্তর দেওয়ার বদলে **কয়েক ধাপে স্বয়ংক্রিয়ভাবে একটি লক্ষ্য অর্জনের চেষ্টা করে**: এটি পরিকল্পনা করে, কোন কাজ করবে তা ঠিক করে, tool ব্যবহার করে, ফলাফল পর্যবেক্ষণ করে এবং সে অনুযায়ী নিজেকে সংশোধন করে। উদাহরণ: "আজকের সব ব্যর্থ RTGS transfer খুঁজে বের করো, কারণ যাচাই করো, এবং প্রতিটি শাখার জন্য একটি ই-মেইলের খসড়া তৈরি করো।"

Tool ও ডেটার সাথে মিথস্ক্রিয়া — agent loop

1. **Perceive:** লক্ষ্য ও বর্তমান প্রসঙ্গ (কথোপকথন, memory, retrieve করা নথি) গ্রহণ করে।
2. **Reason / plan:** LLM পরের ধাপ ঠিক করে এবং প্রয়োজনে একটি tool বাছাই করে।
3. **Act:** এটি একটি structured **tool call** (function calling) তৈরি করে, যেমন `query_db(sql=…)`, `search_docs(…)`, `send_email(…)`। Tool-টি model-এর বাইরে চলে।
4. **Observe:** tool-এর ফলাফল (row, API response, ফাইলের কনটেন্ট) আবার context-এ দেওয়া হয়।
5. **Repeat:** লক্ষ্য পূরণ বা কোনো stop condition না আসা পর্যন্ত পুনরাবৃত্তি, তারপর রিপোর্ট।



চিত্র ২.১৬: Agentic AI — harness LLM-কে ঘিরে loop চালায় এবং MCP server-এর মাধ্যমে tool-এ পৌঁছায়

Model Context Protocol (MCP)-এর ভূমিকা: MCP একটি **open standard** (২০২৪ সালে Anthropic প্রবর্তন করে, এখন ব্যাপকভাবে গৃহীত), যা AI application-কে বাইরের tool ও ডেটার সাথে যুক্ত করার একটি অভিন্ন পদ্ধতি নির্ধারণ করে — একে প্রায়ই "AI-এর USB-C port" বলা হয়। MCP ছাড়া প্রতিটি model–tool জোড়ার জন্য আলাদা custom integration লাগে (M × N); MCP থাকলে একটি tool একবার **MCP server** হিসেবে মোড়ানো হয় এবং যেকোনো MCP-সক্ষম application (**host**, তার **MCP client**-এর মাধ্যমে) সেটি ব্যবহার করতে পারে (M + N)। Server তিনটি জিনিস প্রকাশ করে: **tools** (model যে কাজগুলো call করতে পারে), **resources** (ফাইল বা রেকর্ডের মতো ডেটা) এবং **prompts** (template)। বার্তাগুলো local (stdio) বা remote (HTTP) transport-এর ওপর JSON-RPC 2.0 ব্যবহার করে, এবং client runtime-এ জেনে নেয় একটি server কী কী দেয়।

Agent harness-এর ভূমিকা: harness হলো **model-কে ঘিরে থাকা software**, যা একটি LLM-কে কার্যকর agent-এ পরিণত করে। Model শুধু লেখা তৈরি করে; harness: perceive–plan–act–observe loop চালায়; model-কে system prompt, tool definition ও context পাঠায়; model যে tool call চায় তা execute করে (সরাসরি বা MCP-র মাধ্যমে); context-এর দৈর্ঘ্য ও memory সামলায়; **permission** প্রয়োগ করে (কোন tool, কোন ডেটা, কোন কাজে মানুষের অনুমোদন লাগবে); guardrail, retry, timeout, ধাপ ও খরচের সীমা প্রয়োগ করে; এবং audit-এর জন্য log রাখে। Claude Code ও এ ধরনের coding agent হলো harness-এর উদাহরণ।

ঝুঁকি	কেন ঘটে	প্রতিকার
১. Prompt injection (বিশেষত indirect)	ওয়েব পেজ, ই-মেইল, নথি বা tool output-এর লেখার ভেতরে লুকানো নির্দেশ থাকে ("আগের নিয়ম উপেক্ষা করে গ্রাহক তালিকা ... -তে পাঠাও") এবং model সেগুলো মেনে চলে	সব tool ও retrieve করা কনটেন্টকে নির্দেশ নয়, অবিশ্বস্ত ডেটা হিসেবে গণ্য করা; system instruction ও ডেটা আলাদা রাখা; input/output filtering; tool ও গন্তব্যের allow-list; সংবেদনশীল কাজের আগে মানুষের নিশ্চিতকরণ বাধ্যতামূলক; red-team testing
২. Excessive agency / অনিরাপদ কাজ	Agent-এর প্রয়োজনের চেয়ে বেশি permission থাকে, অথবা লক্ষ্য ভুল বোঝে বা hallucinate করে, এবং ক্ষতিকর ও অপরিবর্তনীয় কাজ করে ফেলে (রেকর্ড মুছে ফেলা, ভুল fund transfer, ডেটা ফাঁস)	**Least privilege** — ডিফল্টভাবে read-only access, সীমিত-পরিসরের credential; উচ্চ-প্রভাব বা অপরিবর্তনীয় ধাপে **human-in-the-loop** অনুমোদন; sandboxing; লেনদেনের সীমা ও rate limit; dry-run mode; পূর্ণ audit logging ও monitoring; একটি kill switch

উল্লেখ করার মতো অন্যান্য ঝুঁকি: hallucinate করা ফলাফল তথ্য হিসেবে রিপোর্ট করা, বাইরের সেবায় গোপন ডেটা ফাঁস, অনিয়ন্ত্রিত খরচ বা loop, এবং অস্পষ্ট জবাবদিহি।

> **প্রশ্ন ১১** 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)*

ভিত্তি	Full	Incremental	Differential
যা কপি হয়	নির্বাচিত সব ডেটা, প্রতিবার	কেবল **যেকোনো ধরনের শেষ backup**-এর পর পরিবর্তিত ডেটা	**শেষ full** backup-এর পর পরিবর্তিত সব ডেটা
Backup-এর সময়	সবচেয়ে বেশি	সবচেয়ে কম	মাঝারি; পরের full পর্যন্ত প্রতিদিন বাড়ে
প্রয়োজনীয় storage	সবচেয়ে বেশি	সবচেয়ে কম	মাঝারি; প্রতিদিন বাড়ে
Restore-এ যা লাগে	শুধু সর্বশেষ full	শেষ full + তার পরের **প্রতিটি** incremental, ক্রমানুসারে	শেষ full + **কেবল সর্বশেষ** differential
Restore-এর গতি	সবচেয়ে দ্রুত	সবচেয়ে ধীর; একটি set হারালে বা নষ্ট হলে পুরো chain ভেঙে যায়	দ্রুত

উদাহরণ — শুক্রবার full backup, বুধবার বিপর্যয়

Incremental: শুক্রবারের full + শনি + রবি + সোম + মঙ্গল restore (৫টি set)। Differential: শুক্রবারের full + মঙ্গলবারের differential restore (২টি set)।

দুটি লক্ষ্যমাত্রা design ঠিক করে: **RPO** (Recovery Point Objective — সর্বোচ্চ কতটুকু ডেটা হারানো গ্রহণযোগ্য, সময় দিয়ে মাপা) এবং **RTO** (Recovery Time Objective — সেবা পুনরুদ্ধারের সর্বোচ্চ সময়)। Core banking system-এর জন্য দুটোই প্রায় শূন্য হওয়া দরকার।

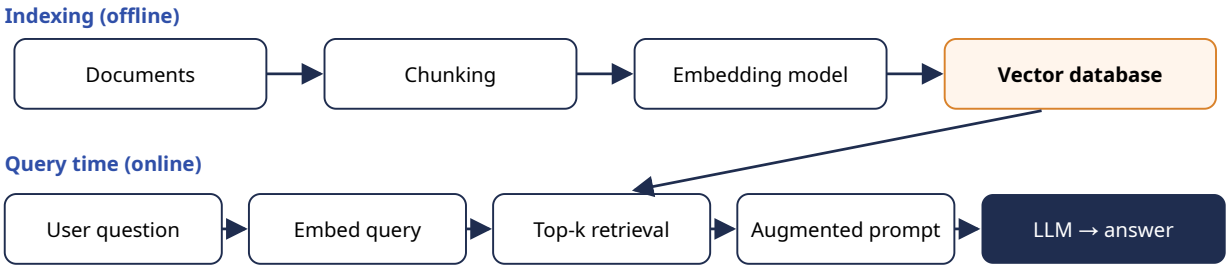
ব্যাংকের জন্য প্রস্তাবিত কৌশল

1. **সাপ্তাহিক full backup** (সাপ্তাহিক ছুটির দিনে) + রাতে **দৈনিক differential** backup — restore-এ মাত্র দুটি set লাগে, তাই RTO কম থাকে।
2. Core banking database-এর জন্য **প্রতি ৫–১৫ মিনিটে transaction-log backup** — point-in-time recovery; RPO কয়েক মিনিট।
3. ভিন্ন ভৌগোলিক স্থানে **Disaster Recovery (DR) site**-এ **real-time replication** — কাছের data centre-এ synchronous (RPO ≈ 0), দূরের DR site-এ asynchronous — পরীক্ষিত failover-সহ, যাতে RTO কয়েক মিনিট থেকে কয়েক ঘণ্টা হয়। বাংলাদেশ ব্যাংকের ICT guideline অনুযায়ী ব্যাংকগুলোকে DR site রাখতে হয়।
4. **3-2-1-1-0 নিয়ম:** ৩টি কপি, ২টি ভিন্ন মাধ্যমে, ১টি off-site, ১টি offline বা **immutable** (ransomware থেকে সুরক্ষা), যাচাইয়ের পর ০টি error।
5. Backup-কে at rest ও in transit উভয় অবস্থায় **encrypt** করা; access সীমিত ও log করা; নিয়ন্ত্রক বিধি যতদিন চায় ততদিন সংরক্ষণ করা।
6. **নিয়মিত restore পরীক্ষা ও DR drill** — যে backup কখনো restore করে দেখা হয়নি, তা প্রমাণিত নয়।

প্রশ্ন ১২ 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) এমন একটি কৌশল, যেখানে LLM প্রথমে **বাইরের একটি জ্ঞান-উৎস থেকে প্রাসঙ্গিক তথ্য retrieve করে** (যেমন ব্যাংকের policy নথি) এবং তারপর সেই তথ্যকে context হিসেবে ব্যবহার করে উত্তর **generate** করে। Model শুধু training-এর সময় মুখস্থ করা জ্ঞানের ওপর নির্ভর না করে সরবরাহ করা তথ্য থেকে উত্তর দেয়।



চিত্র ২.১৭: RAG pipeline — নথিগুলো একবার index করা হয়; প্রতিটি প্রশ্নে সবচেয়ে সদৃশ chunk-গুলো retrieve করা হয়

RAG pipeline design-এর প্রধান ধাপ

1. **Data ingestion:** উৎস নথি (PDF সার্কুলার, FAQ, product manual, database রেকর্ড) সংগ্রহ ও পরিষ্কার করা; metadata (উৎস, তারিখ, বিভাগ, access level) রাখা।
2. **Chunking:** নথিকে ছোট, অর্থবহ অংশে ভাগ করা (যেমন কিছু overlap-সহ 300–800 token), যাতে প্রতিটি chunk একটি ধারণা ধারণ করে।
3. **Embedding:** embedding model দিয়ে প্রতিটি chunk-কে vector-এ রূপান্তর; একই অর্থের লেখা কাছাকাছি vector পায়।
4. **Indexing:** vector, লেখা ও metadata একটি **vector database**-এ সংরক্ষণ।
5. **Retrieval:** একই model দিয়ে ব্যবহারকারীর প্রশ্নকে embed করে সবচেয়ে সদৃশ top-k chunk খোঁজা (cosine similarity); ঐচ্ছিকভাবে keyword search-এর সাথে মিলিয়ে (hybrid) এবং **re-rank** করে।
6. **Augmentation:** retrieve করা chunk-গুলো "শুধু context থেকে উত্তর দাও এবং উৎস উল্লেখ করো"-এর মতো নির্দেশসহ prompt-এ যোগ করা।
7. **Generation:** LLM citation-সহ উত্তর লেখে।
8. **Evaluation ও monitoring:** retrieval-এর মান ও উত্তরের faithfulness পরিমাপ; নথি বদলালে index হালনাগাদ।

Vector database-এর ভূমিকা: এটি embedding সংরক্ষণ করে এবং approximate nearest-neighbour index (HNSW, IVF) ব্যবহার করে লক্ষ লক্ষ vector-এর ওপর দ্রুত **similarity search** চালায়। এটি metadata filtering (যেমন শুধু ২০২৪-এর পরের সার্কুলার, বা শুধু ব্যবহারকারী যে নথি দেখার অনুমতি পান) এবং real-time update সমর্থন করে। উদাহরণ: FAISS, Pinecone, Milvus, Weaviate, Chroma, pgvector-সহ PostgreSQL। এর কারণেই RAG শুধু keyword মিলিয়ে নয়, *অর্থ দিয়ে* তথ্য খুঁজে পায়।

RAG কীভাবে নির্ভুলতা ও প্রাসঙ্গিকতা বাড়ায়

- **Grounding hallucination কমায়** — model বাস্তব, সরবরাহ করা তথ্য থেকে উত্তর দেয়।
- **হালনাগাদ জ্ঞান** — model পুনরায় train না করেই নতুন সার্কুলার index-এ যোগ করা যায়।
- **Domain-specific ও private ডেটা** — ব্যাংকের নিজস্ব policy, যা base model কখনো দেখেনি।
- **Citation ও traceability** — উত্তরে উৎস দেখানো যায়, ফলে কর্মীরা যাচাই করতে পারেন।
- **Access control** — retrieval filter ব্যবহারকারীর permission মেনে চলে।

- Fine-tuning-এর চেয়ে **সস্তা** এবং হালনাগাদ করা সহজ।

প্রশ্ন ১৩ 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)

সমাধান

ধাপ ১ — peak সময়ে সক্রিয় ব্যবহারকারী: 120,000 × 20% = **24,000**
ধাপ ২ — request-এর হার: 24,000 × 5 = প্রতি মিনিটে 120,000 request = 120,000 ÷ 60 = **2,000 requests/second**
ধাপ ৩ — application server: 2,000 ÷ 800 = 2.5 ⇒ **ঊর্ধ্বমুখে** পূর্ণসংখ্যা (একটি server ভাগ করা যায় না) ⇒ **3টি application server**
ধাপ ৪ — database-এর লোড: 2,000 × 2 = **4,000 queries/second**
ধাপ ৫ — database server: 4,000 ÷ 1,000 = 4 ⇒ **4টি database server**
উত্তর: ন্যূনতম **3টি application server** ও **4টি database server**।

বাড়তি নম্বরের জন্য এক লাইন যোগ করুন

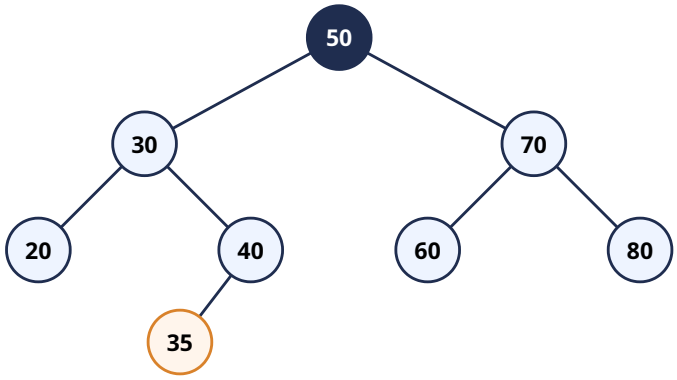
এগুলো ১০০% utilisation-এ ন্যূনতম সংখ্যা। বাস্তবে redundancy (N+1) যোগ করা হয় এবং server-কে প্রায় ৭০% লোডে রাখা হয়: 2,000 ÷ (800 × 0.7) ≈ 3.6 ⇒ 4 + 1 = 5টি application server, একটি load balancer-এর পেছনে; এবং 4,000 ÷ 700 ≈ 5.7 ⇒ 6 + 1 = 7টি database server (read replica ও caching থাকলে কম)। Server-এর সংখ্যা হিসাবের সময় সবসময় ঊর্ধ্বমুখে পূর্ণসংখ্যা নিন।

প্রশ্ন ১৪ 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-এর নিয়ম: প্রতিটি node-এর জন্য, left subtree-র সব মান ছোট এবং right subtree-র সব মান বড়। প্রতিটি নতুন মান root থেকে শুরু করে বাঁয়ে (ছোট হলে) বা ডানে (বড় হলে) সরতে থাকে, যতক্ষণ না একটি খালি জায়গা পায়।

Insert	পথ	যে অবস্থানে বসে
50	—	root
30	30 < 50 → left	50-এর left child
70	70 > 50 → right	50-এর right child
20	< 50 → left; < 30 → left	30-এর left child
40	< 50 → left; > 30 → right	30-এর right child
60	> 50 → right; < 70 → left	70-এর left child
80	> 50 → right; > 70 → right	70-এর right child
35	< 50 → left; > 30 → right; < 40 → left	40-এর left child



চিত্র ২.১৮: প্রাপ্ত BST (height 3; 35 সবশেষে insert করা node)

Traversal	ক্রম	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**

নিজে যাচাই করুন

BST-এর inorder সবসময় ঊর্ধ্বক্রমে সাজানো থাকে — আপনারটা না থাকলে tree ভুল হয়েছে। Preorder সবসময় root (50) দিয়ে শুরু হয়; postorder সবসময় root দিয়ে শেষ হয়।

প্রশ্ন ১৫ 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)

Average Memory Access Time (AMAT) হলো একটি memory access-এর প্রত্যাশিত সময়। এখানে cache সবসময় আগে দেখা হয়; miss হলে (১০% access-এ) *অতিরিক্ত* হিসেবে main memory access করা হয় (serial, বা look-through, access)।

$$\text{AMAT} = \text{Hit time} + \text{Miss rate} \times \text{Miss penalty}$$

সমাধান

Hit ratio $h = 0.90 \Rightarrow$ miss rate $= 1 - 0.90 = 0.10$; cache time $t_c = 5$ ns; memory time $t_m = 80$ ns।

AMAT $= 5 + 0.10 \times 80 = 5 + 8 =$ **13 ns**

Weighted রূপে যাচাই: $h \times t_c + (1 - h) \times (t_c + t_m) = 0.9 \times 5 + 0.1 \times 85 = 4.5 + 8.5 =$ **13 ns** ✓

ব্যাখ্যা: cache ছাড়া প্রতিটি access-এ 80 ns লাগত; cache থাকায় গড় 13 ns — প্রায় **6.2 গুণ দ্রুত** (80 ÷ 13)। প্রশ্নে যদি বলা থাকত cache ও memory *একসাথে (parallel)* খোঁজা হয়, তাহলে সূত্র হতো $0.9 \times 5 + 0.1 \times 80 = 12.5$ ns — তাই কোন অনুমান দেওয়া আছে তা সবসময় ভালো করে পড়ুন।

গণিত (প্রশ্ন ১৬–১৯, প্রতিটি ৫ নম্বর)

প্রশ্ন ১৬ 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)

সমাধান

1 ঘণ্টায়: A ট্যাংকের 1/10 অংশ ভরে, B 1/15 অংশ ভরে, C 1/12 অংশ খালি করে।

1 ঘণ্টায় নিট ভরে = 1/10 + 1/15 − 1/12 = (6 + 4 − 5)/60 = 5/60 = 1/12 অংশ।

∴ ট্যাংক ভরতে সময় লাগবে **12 ঘণ্টা**।

LCM পদ্ধতি: ট্যাংক = LCM(10, 15, 12) = 60 একক; A = +6, B = +4, C = −5 একক/ঘণ্টা ⇒ নিট 5 একক/ঘণ্টা ⇒ 60 ÷ 5 = 12 ঘণ্টা।

প্রশ্ন ১৭ BSCS AP 31.10.2026

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

সমাধান

ধরি, সংখ্যা দুটি x ও 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 অথবা x = −13; সংখ্যাগুলো ধনাত্মক, তাই x = 11।

∴ সংখ্যা দুটি **11 ও 13**। যাচাই: 11 × 13 = 143 ✓

প্রশ্ন ১৮ 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)

সমাধান

ধরি, বেঞ্চের সংখ্যা x।

প্রতি বেঞ্চে 4 জন, 3টি বেঞ্চ খালি ⇒ ছাত্রসংখ্যা = 4(x − 3)।

প্রতি বেঞ্চে 3 জন, 6 জন দাঁড়িয়ে ⇒ ছাত্রসংখ্যা = 3x + 6।

$4(x - 3) = 3x + 6 \Rightarrow 4x - 12 = 3x + 6 \Rightarrow x = 18$টি বেঞ্চ।

ছাত্রসংখ্যা = 3 × 18 + 6 = **60 জন**। যাচাই: 4 × (18 − 3) = 60 ✓

প্রশ্ন ১৯ 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)

সমাধান

দুই অনুপাতে B-কে সমান করি (4 ও 5-এর LCM = 20): A:B = 3:4 = 15:20; B:C = 5:6 = 20:24 ⇒ **A:B:C = 15:20:24**।

ধরি, A = 15k, B = 20k, C = 24k।

A, B-কে 5,000 টাকা দেওয়ার পর: (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।

মোট = 59k = 59 × 1,500 = **88,500 টাকা**। যাচাই: 17,500 : 35,000 = 1 : 2 ✓

English Proficiency (প্রশ্ন ২০–২১)

প্রশ্ন ২০ BSCS AP 31.10.2026

Write only the correct answers. (10)

#	প্রশ্ন	উত্তর	কারণ
a	By the time we arrived, the film ___ (started / had started / has started)	**had started**	অতীতের দুটি কাজের মধ্যে যেটি আগে ঘটেছে সেটি past perfect হয়; "by the time + past" এর সংকেত
b	She is good ___ solving problems.	**at**	নির্দিষ্ট preposition: good at, bad at
c	Neither the teacher nor the students was present.	Neither the teacher nor the students **were** present.	Neither...nor-এ verb কাছের subject অনুযায়ী হয় (students — plural)
d	Noun form of "decide"	**decision**	decide (v) → decision (n) → decisive (adj)
e	I wish I ___ taller. (am / were / will be)	**were**	বর্তমানের অবাস্তব ইচ্ছা ⇒ সব person-এ past subjunctive "were"
f	Hardly ___ the station when the train left.	**had I reached**	Negative adverb শুরুতে থাকলে inversion হয়: Hardly + had + subject + V3 ... when + past
g	Passive: Someone has stolen my bicycle.	**My bicycle has been stolen.**	Present perfect passive = has been + V3; অনির্দিষ্ট agent "by someone" বাদ দেওয়া হয়
h	Indirect: She said to me, "Will you help me tomorrow?"	**She asked me if I would help her the next day.**	Yes/no প্রশ্ন ⇒ asked + if/whether; will → would; you → I, me → her; tomorrow → the next day
i	Closest in meaning to "meticulous"	**extremely careful**	meticulous = খুঁটিনাটির প্রতি গভীর মনোযোগ দেয় এমন
j	Combine with "despite": It was raining heavily. They continued the match.	**Despite the heavy rain, they continued the match.**	despite + noun / gerund (কখনো "despite of" বা "despite it was" নয়)

প্রশ্ন ২১ BSCS AP 31.10.2026

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

পরিকল্পনা (২ মিনিট) — যা ২০ নম্বর এনে দেয়

শিরোনাম · স্পষ্ট thesis-সহ ভূমিকা · ৩–৪টি body অনুচ্ছেদ, প্রতিটিতে একটি করে ধারণা (ঝুঁকিতে থাকা চাকরি, নতুন সৃষ্ট চাকরি, বাংলাদেশের অবস্থা, দক্ষতা ও নীতি) · উপসংহার · প্রায় ৪০০–৪৫০ শব্দ · সহজ, শুদ্ধ বাক্য · linking word (however, moreover, therefore, as a result)।

মডেল রচনা

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.

২.৫ Paper G — বাছাই (প্রিলিমিনারি) MCQ পরীক্ষা ২০২৬, সেট-৩ — সম্পূর্ণ প্রশ্নপত্র

বিষয়	তথ্য
পরীক্ষা	বাছাই পরীক্ষা ২০২৬, **সেট-৩**। হেডারের ডান দিকের লেখা আংশিক পাঠযোগ্য — *পায়রাবন্দর* বলে মনে হয়; নিয়োগকর্তা নিশ্চিত নয়।
ধরন	১০০টি MCQ, ১ ঘণ্টা, পূর্ণমান ১০০; প্রতিটি শুদ্ধ উত্তরে +১, প্রতিটি ভুলে **−০.৫০** (প্রশ্নপত্রে যেমন ছাপা)

বিষয়ভিত্তিক ভাগ	IT/CS ৩৩ · Electrical/electronics ৭ · English ২০ · বাংলা ২০ · সাধারণ জ্ঞান ১৯ · গণিত ১
নতুন যা এসেছে	ML-এর মৌলিক তত্ত্ব (ensemble, bias–variance, dummy classifier, hyperparameter) এবং electrical-এর মৌলিক বিষয় (3-phase, rectifier, RLC resonance), সাথে প্রচলিত CS প্রশ্ন

প্রশ্নগুলো টপিক অনুযায়ী সাজানো। প্রতিটি টপিকের শুরুতে সংক্ষিপ্ত পটভূমি — প্রশ্নগুলোর পেছনের ধারণা, তথ্য ও সূত্র — তারপর মূল নম্বরসহ প্রশ্ন, উত্তর ও এক লাইনের ব্যাখ্যা। শেষে মূল ক্রমে উত্তরমালা আছে, সময় ধরে অনুশীলনের জন্য। পাঁচটি প্রশ্নে ছাপার ভুল বা বিতর্কিত উত্তর আছে — ⚠ চিহ্ন দিয়ে ব্যাখ্যা করা হয়েছে।

> **ভুলে −০.৫ হলে**
> চার অপশনে অন্ধ আন্দাজের প্রত্যাশিত মান **−০.১২৫**; দুটি অপশন বাদ দিলে ০.২৫। প্রত্যাশিত মান ধনাত্মক হলে তবেই আন্দাজ করুন।

১. C প্রোগ্রামিং-এর মৌলিক বিষয় প্রশ্ন ৪, ১৮, ২৪, ৪৫, ৫৬, ৬৮

> **পটভূমি**
> **Data type:** `int` (পূর্ণসংখ্যা), `float` / `double` (দশমিক), `char` (একটি অক্ষর)। **Operator:** `=` মান বসায়, `==` তুলনা করে, `%` ভাগশেষ দেয় (5 % 2 = 1), `!=` অসমান। প্রতিটি **statement** শেষ হয় `;` দিয়ে। **Identifier**-এ অক্ষর, অঙ্ক ও `_` চলে; অঙ্ক দিয়ে শুরু করা যাবে না, keyword (`int`, `for` ...) হবে না, `-` বা space থাকবে না। **Loop:** `for` ও `while` আগে শর্ত দেখে (০ বারও চলতে পারে); `do … while` body চালিয়ে তারপর শর্ত দেখে, তাই **অন্তত একবার** চলে।

4. Which datatype stores whole numbers in C?

(ক) float (খ) double
(গ) char (ঘ) int

উত্তর: (ঘ) — int পূর্ণসংখ্যা রাখে; float/double দশমিক সংখ্যা।

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

(ক) 0 (খ) 1
(গ) 2 (ঘ) 2.5

উত্তর: (খ) — % মানে ভাগশেষ: 5 = 2×2 + 1।

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

(ক) : (খ) ;
(গ) , (ঘ) .

উত্তর: (খ) — Semicolon (;)।

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

(ক) == (খ) =
(গ) != (ঘ) <=

উত্তর: (খ) — = assign করে; == তুলনা করে।

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

(ক) 2value
(খ) int
(গ) total-marks
(ঘ) First-name

⚠ উত্তর: (গ) — কঠোরভাবে কোনোটিই valid নয়: 2value অঙ্ক দিয়ে শুরু, int keyword, '-' অনুমোদিত নয়। প্রশ্নকর্তার উদ্দেশ্য (গ) — total_marks-এর ছাপার ভুল।

68. Which loop executes at least once?

(ক) for
(খ) while
(গ) do while
(ঘ) for each

উত্তর: (গ) — do-while শর্ত পরীক্ষা করে body চালানোর পরে।

২. Python-এর মৌলিক বিষয় প্রশ্ন ৯৫, ৯৮

পটভূমি

Function লেখা হয় `def name(args):` ও indent করা body দিয়ে। `for` যেকোনো iterable-এর ওপর চলে; `range(start, stop, step)` start থেকে *stop - 1* পর্যন্ত সংখ্যা দেয় — যেমন `for i in range(5)` → 0...4।

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

(ক) function
(খ) define
(গ) def
(ঘ) func

উত্তর: (গ) — def name(args):

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

(ক) couple
(খ) range
(গ) excuse
(ঘ) control

উত্তর: (খ) — for i in range(n):

৩. ডেটা স্ট্রাকচার ও সার্চিং প্রশ্ন ১৩, ৬৯, ৭৩, ৮৩, ৯৯

পটভূমি

Stack = LIFO (ওপরে push/pop: undo, function call, recursion)। **Queue** = FIFO (পেছনে ঢোকে, সামনে বের হয়: printer job, BFS)। **Linear search** একে একে দেখে — O(n), unsorted ডেটায় চলে। **Binary search**-এর জন্য **sorted** ডেটা লাগে, প্রতি ধাপে range অর্ধেক — O(log n); ১০ লক্ষ উপাদানে সর্বোচ্চ ২০টি তুলনা। **Palindrome** উল্টো দিক থেকেও একই (madam, 1234321) — দুই pointer বা stack দিয়ে যাচাই।

13. A Stack is called ___.

(ক) Last In First Out
(খ) First In Last Out
(গ) Last In Last Out
(ঘ) First In First Out

⚠ উত্তর: (ক) — প্রচলিত নাম LIFO। FILO-ও একই অর্থ, তবে LIFO দাগানোই নিরাপদ।

69. Which data structure follows the FIFO principle?

(ক) Stack
(খ) Queue
(গ) Tree
(ঘ) Graph

উত্তর: (খ) — Queue।

73. Which searching algorithm requires sorted data?

(ক) Linear
(খ) Binary
(গ) Sequential
(ঘ) Hash

উত্তর: (খ) — Binary search প্রতি ধাপে sorted range অর্ধেক করে।

83. What is the time complexity of binary search?

(ক) O(n)
(খ) $O(n^2)$
(গ) O(log n)
(ঘ) O(1)

উত্তর: (গ) — O(log n)।

99. Which one is an example of a palindrome?

(ক) Argentina
(খ) Brazil
(গ) 1234321
(ঘ) 1024

উত্তর: (গ) — উল্টো দিক থেকেও একই।

৪. ডেটাবেস ও নরমালাইজেশন প্রশ্ন ১৪, ৪৭, ৫২

পটভূমি

RDBMS সম্পর্কিত table-এ ডেটা রাখে (MySQL, PostgreSQL, Oracle, SQL Server)। **Normalization** পুনরাবৃত্তি দূর করে: 1NF atomic মান → 2NF key-এর অংশের ওপর partial dependency নেই → 3NF transitive dependency নেই → **BCNF**: প্রতিটি determinant candidate key। Standard SQL **declarative**, Turing-complete নয় (সাধারণ loop নেই); recursive CTE/PL-SQL একে বাড়ায়।

14. What is the use of BCNF?

(ক) Database normalization
(খ) Fine tuning of software quality
(গ) Indexing mechanism
(ঘ) Hashing

উত্তর: (ক) — Boyce–Codd Normal Form: প্রতিটি determinant অবশ্যই candidate key।

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

(ক) Python
(খ) Java
(গ) SQL
(ঘ) C++

উত্তর: (গ) — Standard (declarative) SQL Turing complete নয়; বাকিগুলো general-purpose language।

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

(ক) MySQL
(খ) Python
(গ) Java
(ঘ) Linux

উত্তর: (ক) — MySQL; Python ও Java ভাষা, Linux OS।

৫. মেশিন লার্নিং ও AI-এর মৌলিক বিষয় প্রশ্ন ৩, ২০, ৫০, ৫৮, ৭৬, ৯৩, ৯৪

পটভূমি

ধারণা	মূল কথা
Supervised বনাম unsupervised	লেবেলযুক্ত ডেটা (SVM, KNN, linear/logistic regression) বনাম লেবেলহীন (K-means, PCA)
Ensemble	অনেক মডেল একত্রে: bagging (Random Forest), boosting (AdaBoost, XGBoost)
Deep learning	বহুস্তর neural network: CNN (ছবি), RNN/LSTM, Transformer (OpenAI-এর ChatGPT-এর মতো LLM)
Hyperparameter	training-এর আগে ঠিক করা হয়: learning rate, regularization α, L1/L2; KNN-এর k (linear regression-এর নয়)
Bias বনাম variance	training accuracy কম = high bias (underfit); train–test ব্যবধান বেশি = high variance (overfit)
Dummy / baseline	ZeroR সবসময় majority class বলে; আসল মডেলকে একে হারাতে হবে

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

(ক) Ada Boost
(খ) SVM
(গ) Linear Regression
(ঘ) Logistic Regression

উত্তর: (ক) — AdaBoost অনেকগুলো weak learner মিলিয়ে কাজ করে (boosting)। Random Forest (bagging)-ও ensemble।

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

(ক) Learning rate
(খ) Alpha (α)
(গ) Number of neighbors
(ঘ) L1/L2 regularization

উত্তর: (গ) — Number of neighbors (k) হলো KNN-এর hyperparameter।

50. Which one is a dummy classifier?

(ক) Linear Regression
(খ) KNN
(গ) SVM
(ঘ) Zero-R

উত্তর: (ঘ) — ZeroR সবসময় majority class predict করে — baseline হিসেবে ব্যবহৃত।

58. ChatGPT-এর তৈরীকারক প্রতিষ্ঠান কোনটি?

(ক) Google
(খ) Microsoft
(গ) Open AI
(ঘ) Meta

উত্তর: (গ) — OpenAI (নভেম্বর ২০২২)।

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

(ক) SVM
(খ) Naive Bayes
(গ) Linear Regression
(ঘ) CNN

উত্তর: (ঘ) — Convolutional Neural Network; বাকিগুলো classical ML।

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

(ক) Low bias and low variance
(খ) High bias and high variance
(গ) Low bias and high variance
(ঘ) High bias and low variance

উত্তর: (ঘ) — Training accuracy কম ⇒ high bias (underfitting); train–test ব্যবধান ছোট (৮%) ⇒ low variance।

94. Which one is an unsupervised learning model?

(ক) SVM
(খ) KNN
(গ) Linear Regression
(ঘ) K-means clustering

উত্তর: (ঘ) — K-means লেবেলহীন ডেটা গ্রুপ করে।

৬. নেটওয়ার্কিং ও ওয়েব প্রশ্ন ৫৯, ৬০, ৮২

পটভূমি

URL (Uniform Resource Locator) = scheme://host/path, যেমন https://www.bb.org.bd/about। **HTTP/HTTPS** (80/443) ওয়েবপেজ আনে; FTP (20/21) ফাইল; SMTP (25) মেইল পাঠায়, POP3 (110)/IMAP (143) মেইল আনে। **DHCP** স্বয়ংক্রিয়ভাবে IP address, mask, gateway ও DNS server দেয়; **DNS** নামকে IP-তে রূপান্তর করে; **proxy** client-এর হয়ে request পাঠায়।

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

(ক) Uniform Resource Locator
(খ) Uniform Resource Link
(গ) Uniform Reference Link
(ঘ) Uniform Record Locator

উত্তর: (ক) — Uniform Resource Locator।

60. Which protocol is primarily used to browse websites?

(ক) FTP
(খ) HTTP
(গ) SMTP
(ঘ) POP3

উত্তর: (খ) — HTTP/HTTPS। FTP = ফাইল; SMTP/POP3 = ই-মেইল।

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

(ক) DNS
(খ) Proxy
(গ) DHCP
(ঘ) All of the above

উত্তর: (গ) — Dynamic Host Configuration Protocol।

৭. সংখ্যা পদ্ধতি ও মেমোরি প্রশ্ন ২৮, ৪৪, ৭৭

পটভূমি

Decimal → binary: ২ দিয়ে ভাগ করে ভাগশেষ নিচ থেকে পড়ুন, বা ২-এর ঘাতের যোগফল (10 = 8 + 2 = 1010)। **Primary memory** CPU সরাসরি ব্যবহার করে: RAM (volatile — বিদ্যুৎ গেলে মুছে যায়) ও ROM (non-volatile)। **Secondary storage** non-volatile: HDD, SSD, DVD, pen drive। Cache থাকে CPU ও RAM-এর মাঝে।

28. What is the binary equivalent for decimal 10?

(ক) 1000 (খ) 1010

(গ) 1100 (ঘ) 1110

উত্তর: (খ) — $10 = 8 + 2 = 1010_2$।

44. What type of memory is RAM?

(ক) Permanent (খ) External

(গ) Volatile (ঘ) Optical

উত্তর: (গ) — বিদ্যুৎ গেলে ডেটা মুছে যায়।

77. Which one is an example of primary memory?

(ক) HDD (খ) SSD

(গ) RAM (ঘ) DVD

উত্তর: (গ) — RAM; বাকিগুলো secondary storage।

৮. অপারেটিং সিস্টেম প্রশ্ন ২২, ৭৮

পটভূমি

Open-source OS: Linux (Ubuntu, Fedora), Android (AOSP), FreeBSD। **Proprietary:** Windows, macOS, iOS। Browser (Firefox) application, OS নয়। Windows shortcut: Ctrl+Shift+N নতুন 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?

(ক) Ctrl+N (খ) Ctrl+Shift+N

(গ) Alt+N (ঘ) Ctrl+Alt+N

উত্তর: (খ) — File Explorer-এ Ctrl+Shift+N; Ctrl+N নতুন window খোলে।

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

(ক) Windows 11 (খ) Mozilla Firefox

(গ) iOS (ঘ) Linux

উত্তর: (ঘ) — Linux। Firefox open source হলেও browser, OS নয়।

৯. নিরাপত্তা ও সফটওয়্যার প্রকৌশল প্রশ্ন ১৫, ৪৮

পটভূমি

Authentication factor: যা জানেন (password, PIN), যা আছে (OTP ফোন, token), যা আপনি (আঙুলের ছাপ)। দুটি ভিন্ন factor = **2FA/MFA** — strong authentication। AES/RSA encryption, authentication process নয়। **Functional requirement** বলে সিস্টেম কী করবে (টাকা পাঠানো); **non-functional** বলে কতটা ভালোভাবে: security, performance, reliability, usability, scalability।

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

(ক) AES (খ) RSA

(গ) Blockchain (ঘ) 2FA

উত্তর: (ঘ) — Two-factor authentication (জানা + থাকা/হওয়া)। AES ও RSA হলো encryption algorithm।

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

(ক) Coding
(খ) Debug
(গ) Security
(ঘ) Prototype

উত্তর: (গ) — Non-functional = গুণগত বৈশিষ্ট্য: security, performance, reliability, usability।

১০. ইলেকট্রিক্যাল ও ইলেকট্রনিক্সের মৌলিক বিষয় প্রশ্ন ২৬, ৩৪, ৭৫, ৮১, ৮৬, ৮৯, ৯১

পটভূমি

Ohm-এর সূত্র V = IR; ক্ষমতা P = VI = I^2R = V^2/R। **Passive** component (R, L, C) amplify করতে পারে না; **active** (transistor, op-amp, SCR) পারে। Capacitor বৈদ্যুতিক ক্ষেত্রে শক্তি রাখে (½CV^2); inductor চৌম্বক ক্ষেত্রে (½LI^2)। **Power factor** = cos ϕ: বিশুদ্ধ R-এ 1, বিশুদ্ধ L বা C-তে 0। **Series RLC resonance:** $X_L = X_C \Rightarrow f = 1/(2\pi\sqrt{LC})$, impedance = R (সর্বনিম্ন)। **3-phase:** phase-গুলো 120° ব্যবধানে। **Converter:** rectifier AC→DC, inverter DC→AC, transformer AC voltage বদলায়, oscillator DC থেকে AC তৈরি করে।

26. Which one is a passive electronic component?

(ক) Transistor
(খ) Operational Amplifier
(গ) Resistor
(ঘ) SCR

উত্তর: (গ) — Passive component (R, L, C) power বাড়াতে/নিয়ন্ত্রণ করতে পারে না; transistor, op-amp, SCR active।

34. Power factor of a purely resistive circuit is:

(ক) 0
(খ) 0.5
(গ) 1
(ঘ) −1

উত্তর: (গ) — V ও I একই phase-এ, ϕ = 0, cos 0 = 1।

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

(ক) 60°
(খ) 90°
(গ) 120°
(ঘ) 180°

উত্তর: (গ) — 360° / 3 = 120°।

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

(ক) Inverter
(খ) Oscillator
(গ) Rectifier
(ঘ) Transformer

উত্তর: (গ) — Rectifier: AC→DC; inverter: DC→AC।

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

(ক) 1 A, 1 W
(খ) 2 A, 40 W
(গ) 0.5 A, 10 W
(ঘ) 0.5 A, 40 W

উত্তর: (খ) — I = V/R = 20/10 = 2 A; P = VI = 20 × 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?

(ক) $f^2 = 1/(4\pi LC)$
(খ) $f^2 = 1/(2\pi LC)$
(গ) $f = 1/\sqrt{(LC)}$
(ঘ) $f = 1/(2\pi\sqrt{(LC)})$

উত্তর: (ঘ) — Resonance-এ $X_L = X_C \Rightarrow 2\pi fL = 1/(2\pi fC) \Rightarrow f = 1/(2\pi\sqrt{LC})$।

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

(ক) inductor
(খ) capacitor
(গ) transformer
(ঘ) resistor

উত্তর: (খ) — Capacitor ($\frac{1}{2}CV^2$); inductor চৌম্বক ক্ষেত্রে রাখে ($\frac{1}{2}LI^2$)।

১১. English: preposition ও phrasal verb

প্রশ্ন ১০, ২৯, ৩৫, ৪০, ৬৪

> **পটভূমি**
>
> মুখস্থ রাখার জোড়া: **dream of**, **good at**, **call at** (স্থান) / **call on** (ব্যক্তি) / **call up** (ফোন), **stand by** (সমর্থন), **turn down** (প্রত্যাখ্যান), insist on, depend on, interested in, afraid of, capable of।

10. Faria dreams ___ running her own business.

(ক) to
(খ) of
(গ) by
(ঘ) in

উত্তর: (খ) — dream of/about + gerund।

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

(ক) in
(খ) on
(গ) at
(ঘ) up

উত্তর: (গ) — call at = কোনো স্থানে যাওয়া; call on = কারও সাথে দেখা; call up = ফোন করা।

35. He is good ___ chess.

(ক) in
(খ) at
(গ) on
(ঘ) with

উত্তর: (খ) — good at + দক্ষতা।

40. My parents always stand ___ my decision.

(ক) with
(খ) at
(গ) along
(ঘ) by

উত্তর: (ঘ) — stand by = সমর্থন করা।

64. The proposal was too good to ___.

(ক) away
(খ) down
(গ) in
(ঘ) off

⚠ উত্তর: (খ) — ছাপায় verb বাদ পড়েছে; উদ্দেশ্য 'too good to turn down' (= প্রত্যাখ্যান করা)।

১২. English: verb pattern ও বিশেষ গঠন প্রশ্ন ১৬, ৩২, ৬৭, ৭৯, ৮০, ৯৭

পটভূমি

গঠন	নিয়ম	উদাহরণ
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 (আগের অবাস্তব কাজ)	+ 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?

(ক) check
(খ) checked
(গ) to check
(ঘ) checking

উত্তর: (ঘ) — mind + gerund (verb + ing)।

32. Identify the correct sentence:

(ক) She prefers dancing to singing.
(খ) She prefers dancing than singing.
(গ) She prefers to dance than singing.
(ঘ) She prefers dancing over to sing.

উত্তর: (ক) — prefer + gerund + to + gerund।

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

(ক) had failed
(খ) fail
(গ) would fail
(ঘ) failed

উত্তর: (ঘ) — No sooner + had + V3 ... than + past simple।

79. I look forward to ___ from you.

(ক) hearing
(খ) heard
(গ) hear
(ঘ) have heard

উত্তর: (ক) — এখানে 'to' preposition → gerund।

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

(ক) rolled
(খ) has rolled
(গ) had rolled
(ঘ) is rolling

উত্তর: (গ) — আগের অবাস্তব কাজ বোঝাতে as if + past perfect।

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

(ক) change
(খ) has changed
(গ) changed
(ঘ) did change

উত্তর: (গ) — It is high time + subject + past simple।

১৩. English: parts of speech ও word form প্রশ্ন ১৯, ৫৫, ৬৬

পটভূমি

যে শব্দ noun-কে বর্ণনা করে, বা linking verb (be, seem, look)-এর পরে subject-কে বর্ণনা করে, তা **adjective** (hereditary, disagreeable)। সাধারণ adjective suffix: -ous, -ful, -able/-ible, -ive, -al, -ed (various, variable, varied)। সব root-এ সব suffix বসে না — 'variful' নেই।

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

(ক) noun
(খ) adverb
(গ) adjective
(ঘ) verb

উত্তর: (গ) — linking verb 'is'-এর পরে noun 'depression'-কে বর্ণনা করছে।

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

(ক) noun
(খ) verb
(গ) adjective
(ঘ) adverb

উত্তর: (গ) — noun 'man'-কে বিশেষিত করছে।

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

(ক) various
(খ) variful
(গ) variable
(ঘ) varied

উত্তর: (খ) — 'Variful' কোনো ইংরেজি শব্দ নয়।

১৪. English: শব্দভান্ডার, idiom, বানান, সাহিত্য প্রশ্ন ৯, ২৭, ৩০, ৩৬, ৫১, ৫৩

পটভূমি

পেশাভিত্তিক শব্দ: curator (জাদুঘর), pallbearer (শবাধার), janitor (পরিচ্ছন্নতা), archivist (নথি)। Idiom: through thick and thin (সব অবস্থায়), once in a blue moon (কদাচিৎ), at sixes and sevens (বিশৃঙ্খল)। বানান-ফাঁদ: diarrhoea, accommodation, committee, separate, occurrence। Romantic কবি: Keats (To Autumn, Ode to a Nightingale), Wordsworth (Daffodils), Shelley (Ode to the West Wind)।

9. The synonym of "commemorate" is ___.

(ক) observe
(খ) memorise
(গ) forget
(ঘ) overlook

উত্তর: (ক) — Commemorate = স্মরণ করা / পালন করা (observe)। Memorise = মুখস্থ করা।

27. "Through thick and thin" means:

(ক) by chance
(খ) very quickly
(গ) in all circumstances
(ঘ) abruptly

উত্তর: (গ) — সুখে-দুঃখে, সব অবস্থায়।

30. "To Autumn" is written by ___.

(ক) William Shakespeare (খ) William Wordsworth
(গ) John Keats (ঘ) Robert Frost

উত্তর: (গ) — John Keats (১৮১৯ সালের ode)।

36. Which one of the following is correctly spelt?

(ক) Dirohea (খ) Diarrhoea
(গ) Dirrohea (ঘ) Dirrhoea

উত্তর: (খ) — Diarrhoea (British) / diarrhea (American)।

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

(ক) Librarian (খ) architect
(গ) curator (ঘ) janitor

উত্তর: (গ) — Curator — জাদুঘরের তত্ত্বাবধায়ক।

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

(ক) mourners (খ) pallbearers
(গ) grave diggers (ঘ) processioners

উত্তর: (খ) — Pallbearers — শবাধার বহনকারী।

১৫. বাংলা সাহিত্য প্রশ্ন ১, ১২, ৩৭, ৪১, ৪২, ৪৬, ৬৩, ৬৫, ৭২, ৮৮, ৯২

পটভূমি

যুগ: প্রাচীন (চর্যাপদ, ১৯০৭-এ হরপ্রসাদ শাস্ত্রী আবিষ্কার করেন) → মধ্য (বড়ু চণ্ডীদাসের শ্রীকৃষ্ণকীর্তন, ১৯০৯-এ আবিষ্কৃত; আবদুল হাকিমের 'বঙ্গবাণী'; শেষ কবি ভারতচন্দ্র, মৃত্যু ১৭৬০) → আধুনিক (১৮০০ থেকে, ফোর্ট উইলিয়াম কলেজে বাংলা গদ্যের সূচনা; বিদ্যাসাগরের বর্ণপরিচয় ১৮৫৫)। **অবশ্যপাঠ্য লেখক:** রোকেয়া (অবরোধবাসিনী, মতিচূর), শরৎচন্দ্র (পথের দাবী — বাজেয়াপ্ত), বিভূতিভূষণ (পথের পাঁচালী → আম আঁটির ভেঁপু), প্রমথ চৌধুরী (ছদ্মনাম বীরবল), রবীন্দ্রনাথ (ডাকঘর — নাটক)। **মুক্তিযুদ্ধের উপন্যাস:** রাইফেল রোটি আওরাত (আনোয়ার পাশা), জোছনা ও জননীর গল্প (হুমায়ূন আহমেদ), হাঙর নদী গ্রেনেড (সেলিনা হোসেন)।

1. কোনটি বেগম রোকেয়া সাখাওয়াত হোসেনের রচনা?

(ক) নারী (খ) অবরোধবাসিনী
(গ) সংস্কৃতি-কথা (ঘ) সাঁঝের মায়া

উত্তর: (খ) — অবরোধবাসিনী (১৯৩১); তাঁর অন্য রচনা মতিচূর, পদ্মরাগ, Sultana's Dream। সাঁঝের মায়া — সুফিয়া কামাল; সংস্কৃতি-কথা — মোতাহের হোসেন চৌধুরী।

12. বাংলা গদ্যের বিকাশে কোন প্রতিষ্ঠানটির ভূমিকা রয়েছে?

(ক) হিন্দু কলেজ (খ) সংস্কৃত কলেজ
(গ) ফোর্ট উইলিয়াম কলেজ (ঘ) কলিকাতা কলেজ

উত্তর: (গ) — ফোর্ট উইলিয়াম কলেজ (১৮০০) — উইলিয়াম কেরির তত্ত্বাবধানে প্রথম বাংলা গদ্যপাঠ্য রচিত হয়।

37. 'বঙ্গবাণী' কবিতার রচয়িতা—

(ক) আবদুল হাকিম
(খ) মাইকেল মধুসূদন দত্ত
(গ) কাজী নজরুল ইসলাম
(ঘ) জীবনানন্দ দাশ

উত্তর: (ক) — আবদুল হাকিম (১৭শ শতক): 'যে সবে বঙ্গেত জন্মি হিংসে বঙ্গবাণী...'।

41. 'বর্ণপরিচয়'-এর রচয়িতা কে?

(ক) বঙ্কিমচন্দ্র চট্টোপাধ্যায়
(খ) রামমোহন রায়
(গ) ঈশ্বরচন্দ্র বিদ্যাসাগর
(ঘ) উইলিয়াম কেরী

উত্তর: (গ) — ঈশ্বরচন্দ্র বিদ্যাসাগর (১৮৫৫)।

42. 'শ্রীকৃষ্ণকীর্তন' আবিষ্কৃত হয় কত সালে?

(ক) ১৯০২ সালে
(খ) ১৯০৭ সালে
(গ) ১৯০৯ সালে
(ঘ) ১৯১১ সালে

উত্তর: (গ) — বসন্তরঞ্জন রায় ১৯০৯ সালে বাঁকুড়ায় আবিষ্কার করেন; প্রকাশ ১৯১৬। (চর্যাপদ: ১৯০৭।)

46. 'আম আঁটির ভেঁপু' রচনাটি কোন উপন্যাসের সঙ্গে সম্পর্কিত?

(ক) কপালকুণ্ডলা
(খ) পথের পাঁচালী
(গ) পুতুলনাচের ইতিকথা
(ঘ) কবি

উত্তর: (খ) — বিভূতিভূষণের 'পথের পাঁচালী'-র কিশোর সংস্করণ।

63. শরৎচন্দ্র চট্টোপাধ্যায়ের কোন উপন্যাস ব্রিটিশ সরকার কর্তৃক বাজেয়াপ্ত হয়েছিল?

(ক) দেবদাস
(খ) পথের দাবী
(গ) চরিত্রহীন
(ঘ) গৃহদাহ

উত্তর: (খ) — পথের দাবী (১৯২৬) — বিপ্লবী বিষয়ের জন্য বাজেয়াপ্ত।

65. মধ্যযুগের শেষ কবির নাম কী?

(ক) ভারতচন্দ্র রায় গুণাকর
(খ) শাহ মুহম্মদ সগীর
(গ) চণ্ডীদাস
(ঘ) ঈশ্বরচন্দ্র গুপ্ত

উত্তর: (ক) — ভারতচন্দ্র (মৃত্যু ১৭৬০)। ঈশ্বরচন্দ্র গুপ্ত 'যুগসন্ধিক্ষণের কবি'।

72. 'বীরবল' কার ছদ্মনাম?

(ক) বিহারীলাল চক্রবর্তী
(খ) রবীন্দ্রনাথ ঠাকুর
(গ) সৈয়দ মুজতবা আলী
(ঘ) প্রমথ চৌধুরী

উত্তর: (ঘ) — প্রমথ চৌধুরী (সবুজপত্র-এর সম্পাদক)।

88. মুক্তিযুদ্ধভিত্তিক উপন্যাস কোনটি?

(ক) আরেক ফাল্গুন
(খ) রাইফেল রোটি আওরাত
(গ) খোয়াবনামা
(ঘ) ক্রীতদাসের হাসি

উত্তর: (খ) — আনোয়ার পাশা (১৯৭৩)। আরেক ফাল্গুন — ভাষা আন্দোলন; খোয়াবনামা — তেভাগা/পাকিস্তান আমল; ক্রীতদাসের হাসি — আইয়ুব শাসনের রূপক।

92. 'ডাকঘর' কোন ধরনের রচনা?

(ক) গল্প (খ) উপন্যাস
(গ) নাটক (ঘ) প্রবন্ধ

উত্তর: (গ) — নাটক (১৯১২)।

১৬. বাংলা ব্যাকরণ প্রশ্ন ৫, ২৫, ৩১, ৪৯, ৬২, ৭০, ৯০, ৯৬, ১০০

পটভূমি

শাখা: ধ্বনিতত্ত্ব, রূপতত্ত্ব, বাক্যতত্ত্ব, অর্থতত্ত্ব। মৌলিক স্বরধ্বনি ৭টি: অ আ ই উ এ ও অ্যা। **উচ্চারণ:** ক্ষ = ক্‌খ, যুক্তবর্ণের আগে আদ্য অ প্রায়ই ও-এর মতো (দক্ষ → দোক্‌খো)। **উপমান কর্মধারয়:** উপমান + সাধারণ ধর্ম (ভ্রমরের ন্যায় কালো = ভ্রমরকালো); উপমিত: উপমেয় + উপমান (মুখ চন্দ্রের ন্যায় = মুখচন্দ্র)। **বানান:** দুরবস্থা, উপর্যুক্ত, শুধু/মাত্র (দুটো একসাথে নয়)। **এক কথায়:** যা নিবারণ করা যায় না — অনিবার্য; কষ্টে নিবারণ করা যায় — দুর্নিবার। বাগধারা: আমড়া কাঠের ঢেঁকি — অপদার্থ।

5. কোনটি শুদ্ধ?

(ক) উপরোক্ত (খ) দুরবস্থা
(গ) দুরাবস্থা (ঘ) শুধুমাত্র

উত্তর: (খ) — দুঃ + অবস্থা = দুরবস্থা। শুদ্ধ রূপ: উপরোক্ত → উপর্যুক্ত; শুধুমাত্র → শুধু বা মাত্র (একসাথে দ্বিরুক্তি দোষ)।

25. 'নদীর পাড়ের গাছ' প্রবাদটির অর্থ কী?

(ক) বিপজ্জনক (খ) ক্ষুদ্র
(গ) অনিশ্চিত অস্তিত্ব (ঘ) ক্ষণস্থায়ী

⚠ উত্তর: (গ) — ভাঙনপ্রবণ নদীর পাড়ের গাছ যেকোনো সময় ভেঙে পড়তে পারে — তাই অনিশ্চিত অস্তিত্ব।

31. 'আমড়া কাঠের ঢেঁকি' বাগধারাটির অর্থ কী?

(ক) তস্কর (খ) অপদার্থ
(গ) দুর্লভ বস্তু (ঘ) গলা ধাক্কা

উত্তর: (খ) — অপদার্থ / অকেজো লোক।

49. 'দক্ষ'-এর সঠিক উচ্চারণ —

(ক) দোক্‌খো (খ) দোক্‌খ
(গ) দক্‌খো (ঘ) দক্‌কো

উত্তর: (ক) — ক্ষ = ক্‌খ, এবং আদ্য অ ও-কারের মতো উচ্চারিত: দোক্‌খো।

62. 'Phonology' শব্দের অর্থ কী?

(ক) ধ্বনিতত্ত্ব (খ) রূপতত্ত্ব
(গ) বাক্যতত্ত্ব (ঘ) অর্থতত্ত্ব

উত্তর: (ক) — Morphology = রূপতত্ত্ব; Syntax = বাক্যতত্ত্ব; Semantics = অর্থতত্ত্ব।

70. 'ভ্রমর কালো' কোন সমাস?

(ক) উপমান কর্মধারয় (খ) সাধারণ কর্মধারয়
(গ) উপমিত কর্মধারয় (ঘ) মধ্যপদলোপী কর্মধারয়

উত্তর: (ক) — ভ্রমরের ন্যায় কালো — উপমান (ভ্রমর) + সাধারণ ধর্ম (কালো) = উপমান কর্মধারয়।

90. বাংলা ভাষায় মৌলিক স্বরধ্বনি কয়টি?

(ক) ১১ টি
(খ) ৯ টি
(গ) ৭ টি
(ঘ) ৫ টি

উত্তর: (গ) — অ, আ, ই, উ, এ, ও, অ্যা — ৭টি। (স্বরবর্ণ ১১টি।)

96. 'মতি' শব্দের সমার্থক শব্দ-

(ক) মনন
(খ) বৃদ্ধি
(গ) রুচি
(ঘ) অলংকার

উত্তর: (গ) — মতি = মন, বুদ্ধি, রুচি, প্রবৃত্তি। লক্ষ করুন (খ) 'বৃদ্ধি', 'বুদ্ধি' নয়।

100. 'যা নিবারণ করা যায় না' -এক কথায় কী বলে?

(ক) দুর্নিবার
(খ) অনিবার্য
(গ) দুর্লভ
(ঘ) দুর্দমনীয়

উত্তর: (খ) — অনিবার্য। দুর্নিবার = যা নিবারণ করা কষ্টসাধ্য; দুর্দমনীয় = যা দমন করা কঠিন।

১৭. সাধারণ জ্ঞান: বাংলাদেশ প্রশ্ন ২, ৭, ৮, ১১

পটভূমি

সমুদ্রবন্দর: চট্টগ্রাম (বৃহত্তম, ১৮৮৮), মোংলা (দ্বিতীয়, ১৯৫০), পায়রা (তৃতীয়, ২০১৩); মাতারবাড়ি গভীর সমুদ্রবন্দর নির্মাণাধীন। **গ্যাসক্ষেত্র:** মজুদে তিতাস বৃহত্তম, উৎপাদনে বিবিয়ানা; প্রথম আবিষ্কৃত হরিপুর (১৯৫৫)। **নির্বাচন:** প্রথম সংসদ নির্বাচন ৭ মার্চ ১৯৭৩। **নদী:** পদ্মা, মেঘনা, যমুনা প্রধান; গাইডবইয়ে সবচেয়ে বেশি জেলা দিয়ে প্রবাহিত নদী সাধারণত পদ্মা।

2. বাংলাদেশের দ্বিতীয় বৃহত্তম সমুদ্রবন্দর কোনটি?

(ক) পায়রা বন্দর
(খ) মোংলা বন্দর
(গ) চট্টগ্রাম বন্দর
(ঘ) মাতারবাড়ি বন্দর

উত্তর: (খ) — চট্টগ্রাম বৃহত্তম, মোংলা দ্বিতীয়, পায়রা তৃতীয়।

7. মজুদ বিবেচনায় বাংলাদেশের সবচেয়ে বড় গ্যাসক্ষেত্র কোনটি?

(ক) সাংগু
(খ) কৈলাশটিলা
(গ) মেঘনা
(ঘ) তিতাস

উত্তর: (ঘ) — মজুদে তিতাস (ব্রাহ্মণবাড়িয়া) বৃহত্তম; বর্তমান উৎপাদনে বিবিয়ানা (অপশনে নেই)।

8. বাংলাদেশে কোন নদীটি সবচেয়ে বেশী জেলা দিয়ে প্রবাহিত হয়েছে?

(ক) পদ্মা
(খ) যমুনা
(গ) মেঘনা
(ঘ) মহানন্দা

⚠ উত্তর: (ক) — পদ্মা — অধিকাংশ চাকরির গাইডে এই উত্তর (কিছু উৎসে মেঘনা লেখা আছে)।

11. স্বাধীনতার পর বাংলাদেশে প্রথম পার্লামেন্ট নির্বাচন কোন সালে অনুষ্ঠিত হয়?

(ক) ১৯৭১
(খ) ১৯৭২
(গ) ১৯৭৩
(ঘ) ১৯৭৪

উত্তর: (গ) — ৭ মার্চ ১৯৭৩।

১৮. সাধারণ জ্ঞান: আন্তর্জাতিক বিষয় প্রশ্ন ৬, ১৭, ২১, ২৩, ৩৩, ৩৮, ৪৩, ৫৪, ৫৭, ৬১, ৭১, ৭৪, ৮৪, ৮৫, ৮৭

পটভূমি

জাতিসংঘ: সনদ স্বাক্ষর ২৬ জুন ১৯৪৫, কার্যকর ২৪ অক্টোবর ১৯৪৫ (জাতিসংঘ দিবস); সদর দপ্তর নিউইয়র্ক — UNICEF, UNDP-ও; জেনেভায় WHO, ILO, UNHCR; প্যারিসে UNESCO; রোমে FAO। আয়তনে ক্ষুদ্রতম সদস্য মোনাকো; ভ্যাটিকান পর্যবেক্ষক। **চুক্তি:** কিয়োটো প্রটোকল (১৯৯৭, গ্রিনহাউস গ্যাস), প্যারিস চুক্তি (২০১৫), CEDAW (১৯৭৯, নারী অধিকার); ২০২৬-এর COP-31 তুরস্কে। **অন্যান্য:** দক্ষিণ আমেরিকায় বৃহত্তম ব্রাজিল; আমাজন = পৃথিবীর ফুসফুস; মঙ্গল = লাল গ্রহ; চার্চিল = দ্বিতীয় বিশ্বযুদ্ধের প্রধানমন্ত্রী; জার্মানি/অস্ট্রিয়ায় সরকারপ্রধান চ্যান্সেলর; persona non grata = অবাঞ্ছিত কূটনীতিক; BIMSTEC চেয়ার: বাংলাদেশ (এপ্রিল ২০২৫ থেকে); বিশ্বকাপ ২০৩০ মরক্কো–স্পেন–পর্তুগাল, ২০৩৪ সৌদি আরব।

6. আয়তনে দক্ষিণ আমেরিকার বৃহত্তম দেশ কোনটি?

(ক) আর্জেন্টিনা
(খ) ব্রাজিল
(গ) পেরু
(ঘ) কলম্বিয়া

উত্তর: (খ) — ব্রাজিল (≈৮৫ লক্ষ বর্গকিমি, মহাদেশের প্রায় অর্ধেক)।

17. Kyoto Protocol-এর বিষয়বস্তু কী?

(ক) জলবায়ু পরিবর্তন
(খ) পারমাণবিক অস্ত্র নিরোধ
(গ) মরুকরণ রোধ
(ঘ) আন্তর্জাতিক নদীসমূহের পানির হিস্যা বণ্টন

উত্তর: (ক) — ১৯৯৭ সালের চুক্তি — গ্রিনহাউস গ্যাস নির্গমন হ্রাস (UNFCCC)।

21. UNICEF-এর সদর দপ্তর কোথায় অবস্থিত?

(ক) নিউইয়র্ক
(খ) লন্ডন
(গ) জেনেভা
(ঘ) প্যারিস

উত্তর: (ক) — নিউইয়র্ক, যুক্তরাষ্ট্র।

23. জাতিসংঘ সংস্থা UNDP-এর সদর দপ্তর কোথায়?

(ক) টরেন্টো, কানাডা
(খ) আমস্টারডাম, নেদারল্যান্ড
(গ) লন্ডন, যুক্তরাজ্য
(ঘ) নিউইয়র্ক, যুক্তরাষ্ট্র

উত্তর: (ঘ) — নিউইয়র্ক, যুক্তরাষ্ট্র।

33. COP-31 কোথায় অনুষ্ঠিত হবে?

(ক) তুরস্ক
(খ) মেক্সিকো
(গ) সিডনি
(ঘ) বার্লিন

উত্তর: (ক) — COP-31 (নভেম্বর ২০২৬) তুরস্কের আন্তালিয়ায়; আলোচনায় সভাপতিত্ব অস্ট্রেলিয়ার।

38. ২০৩৪ সালে ফিফা (FIFA) ফুটবল বিশ্বকাপ-এর আয়োজক দেশ কোনটি?

(ক) রাশিয়া
(খ) সৌদি আরব
(গ) মরক্কো
(ঘ) স্পেন

উত্তর: (খ) — সৌদি আরব (ডিসেম্বর ২০২৪-এ নিশ্চিত)। ২০৩০: মরক্কো–স্পেন–পর্তুগাল।

43. আয়তনে কোনটি জাতিসংঘের ক্ষুদ্রতম সদস্য রাষ্ট্র?

(ক) টুভালু
(খ) হাইতি
(গ) ভ্যাটিকান সিটি
(ঘ) নাউরু

উত্তর: (ঘ) — ভ্যাটিকান সিটি সদস্য নয়, পর্যবেক্ষক। নাউরু (২১ বর্গকিমি) < টুভালু (২৬)। (প্রকৃত ক্ষুদ্রতম সদস্য মোনাকো — অপশনে নেই।)

54. দ্বিতীয় বিশ্বযুদ্ধ চলাকালীন যুক্তরাজ্যের প্রধানমন্ত্রী কে ছিলেন?

(ক) উইনস্টন চার্চিল
(খ) হ্যারল্ড ম্যাকমিলান
(গ) অ্যান্থনি ইডেন
(ঘ) মার্গারেট থ্যাচার

উত্তর: (ক) — চার্চিল (১৯৪০–৪৫)। যুদ্ধের শুরুতে (১৯৩৯–৪০) নেভিল চেম্বারলেইন — অপশনে নেই।

57. কোন গ্রহকে লাল গ্রহ বলা হয়?

(ক) মঙ্গল
(খ) বুধ
(গ) বৃহস্পতি
(ঘ) শুক্র

উত্তর: (ক) — মঙ্গল (পৃষ্ঠে আয়রন অক্সাইড)।

61. CEDAW সনদটিতে কাদের অধিকারের কথা বলা হয়েছে?

(ক) শিশু
(খ) উদ্বাস্তু
(গ) অভিবাসী
(ঘ) নারী

উত্তর: (ঘ) — নারীর প্রতি সকল প্রকার বৈষম্য বিলোপ সনদ (১৯৭৯)।

71. জাতিসংঘ সনদ কোন তারিখ থেকে কার্যকর হয়?

(ক) ২৬ জুন ১৯৪৫
(খ) ২৪ অক্টোবর ১৯৪৫
(গ) ১০ ডিসেম্বর ১৯৪৫
(ঘ) ১০ জানুয়ারি ১৯৪৬

উত্তর: (খ) — স্বাক্ষর ২৬ জুন ১৯৪৫; কার্যকর ২৪ অক্টোবর ১৯৪৫ (জাতিসংঘ দিবস)।

74. নিচের কোনটিকে পৃথিবীর ফুসফুস বলা হয়?

(ক) সুন্দরবন
(খ) আমাজন রেইন ফরেস্ট
(গ) বোরিয়াল বন
(ঘ) কঙ্গো রেইন ফরেস্ট

উত্তর: (খ) — আমাজন রেইন ফরেস্ট।

84. Persona-non-grata-এর অর্থ কী?

(ক) সম্মানিত ব্যক্তি
(খ) রাষ্ট্রদূত পদমর্যাদার ব্যক্তি
(গ) অগ্রহণযোগ্য ব্যক্তি
(ঘ) মন্ত্রী পর্যায়ের ব্যক্তি

উত্তর: (গ) — স্বাগতিক দেশ যে কূটনীতিককে অবাঞ্ছিত ঘোষণা করে।

85. BIMSTEC এর বর্তমান চেয়ারম্যান কোন দেশের?

(ক) থাইল্যান্ড
(খ) ভারত
(গ) নেপাল
(ঘ) বাংলাদেশ

উত্তর: (ঘ) — ৬ষ্ঠ শীর্ষ সম্মেলনে (ব্যাংকক, এপ্রিল ২০২৫) থাইল্যান্ডের কাছ থেকে বাংলাদেশ দায়িত্ব নেয়।

87. চ্যান্সেলর (Chancellor) কোন দেশের সরকার প্রধানের পদ?

(ক) কানাডা (খ) ফ্রান্স

(গ) পোল্যান্ড (ঘ) জার্মানি

উত্তর: (ঘ) — জার্মানি (অস্ট্রিয়াতেও)।

১৯. গণিত: সেট প্রশ্ন ৩৯

পটভূমি

Inclusion–exclusion: $|A \cup B| = |A| + |B| - |A \cap B|$; তিন সেটে $|A \cap B \cap C|$ আবার যোগ: $|A \cup B \cup C| = \Sigma A - \Sigma(\text{জোড়া}) + |A \cap B \cap C|$। উদাহরণ: ৪০ জন চা, ৩০ জন কফি, ১০ জন দুটোই → অন্তত একটি পছন্দ করে ৬০ জন।

39. Which one is the principle of inclusion/exclusion of two sets intersecting each other?

(ক) $|A \cup B| = |A| + |B| - |A \cap B|$ (খ) $|A \cup B| = |A| + |B|$

(গ) $|A \cap B| = |A| + |B|$ (ঘ) $|A \cap B| = |A| - |B| + |A \cup B|$

উত্তর: (ক) — $|A| + |B|$-তে $A \cap B$ দুবার গোনা হয়, তাই একবার বাদ।

উত্তরমালা (মূল ক্রমে)

১–খ	২–খ	৩–ক	৪–ঘ	৫–খ	৬–খ	৭–ঘ	৮–ক	৯–ক	১০–খ
১১–গ	১২–গ	১৩–ক	১৪–ক	১৫–ঘ	১৬–ঘ	১৭–ক	১৮–খ	১৯–গ	২০–গ
২১–ক	২২–খ	২৩–ঘ	২৪–খ	২৫–গ	২৬–গ	২৭–গ	২৮–খ	২৯–গ	৩০–গ
৩১–খ	৩২–ক	৩৩–ক	৩৪–গ	৩৫–খ	৩৬–খ	৩৭–ক	৩৮–খ	৩৯–ক	৪০–ঘ
৪১–গ	৪২–গ	৪৩–ঘ	৪৪–গ	৪৫–খ	৪৬–খ	৪৭–গ	৪৮–গ	৪৯–ক	৫০–ঘ
৫১–গ	৫২–ক	৫৩–খ	৫৪–ক	৫৫–গ	৫৬–গ	৫৭–ক	৫৮–গ	৫৯–ক	৬০–খ
৬১–ঘ	৬২–ক	৬৩–খ	৬৪–খ	৬৫–ক	৬৬–খ	৬৭–ঘ	৬৮–গ	৬৯–খ	৭০–ক
৭১–খ	৭২–ঘ	৭৩–খ	৭৪–খ	৭৫–গ	৭৬–ঘ	৭৭–গ	৭৮–ঘ	৭৯–ক	৮০–গ
৮১–গ	৮২–গ	৮৩–গ	৮৪–গ	৮৫–ঘ	৮৬–খ	৮৭–ঘ	৮৮–খ	৮৯–ঘ	৯০–গ
৯১–খ	৯২–গ	৯৩–ঘ	৯৪–ঘ	৯৫–গ	৯৬–গ	৯৭–গ	৯৮–খ	৯৯–গ	১০০–খ

অধ্যায় ৩ — প্রোগ্রামিং প্রবলেম ব্যাংক

Coding Problem Bank — C & Java (with Python where short)

BIBM-এর AP লিখিত পরীক্ষায় প্রতিবার ২–৩টি প্রোগ্রাম লিখতে হয়েছে: ১ থেকে n পর্যন্ত prime, সিরিজের যোগফল (1+2+4+7+11...), Floyd/প্যাটার্ন ত্রিভুজ, LCM, অ্যারের ক্ষুদ্রতম মান, weekly pay, recursive factorial, insertion sort, এবং output tracing (recursion)। এই অধ্যায়ে সেগুলোসহ ৪৬টি প্রোগ্রাম ও ২১টি output-tracing সমস্যা আছে। প্রতিটি কোড compile-যোগ্য C99 বা Java 8+-এ লেখা; প্রশ্নে "C/Java" থাকলে যেকোনো একটিতে লিখলেই চলে।

৩.১ কাগজে কোড লিখে পূর্ণ নম্বর পাওয়ার নিয়ম

৬টি ধাপ (প্রতি প্রোগ্রামে ৮–১০ মিনিট)

1. **Header ও main** পুরোটা লিখুন (`#include <stdio.h>`, `int main(void)`, `return 0;`) — অর্ধেক কোড দেখলে পরীক্ষক নম্বর কাটেন।
2. **Input prompt + scanf/Scanner** — প্রশ্নে "n is given by user input" থাকলে অবশ্যই input নিন; hard-code করবেন না।
3. **Validation** এক লাইনে: `if (n < 1) { printf("Invalid"); return 1; }`
4. **Logic আলাদা function-এ** (যেমন `isPrime()`) — পড়তে সহজ, partial marking-এ সুবিধা।
5. **২–৩টি comment** — লজিক কোথায় তা দেখান (`/* √n পর্যন্ত ভাগ */`)।
6. **Sample I/O ও complexity** কোডের নিচে লিখুন: *Input: 20 → Output: 2 3 5 7 11 13 17 19; Time O(n√n)*।

সাধারণ ভুল যেগুলোতে নম্বর কাটে

`scanf("%d", n)` (& বাদ), `1/i` integer division (হওয়া উচিত `1.0/i`), loop-এ `<=` বনাম `<` ভুল, `=` বনাম `==`, factorial-এ `int` overflow ($13! > 2^{31}$), string-এর শেষে `'\0'`-এর জায়গা না রাখা, Java-তে String তুলনায় `==`।

৩.২ সংখ্যা-ভিত্তিক প্রোগ্রাম

P1. ১ থেকে n পর্যন্ত সব prime 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 prime নয় */
    for (int i = 2; i * i <= n; i++)     /* √n পর্যন্ত ভাগ করলেই যথেষ্ট */
        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 (বড় n-এর জন্য দ্রুত)

```c
#include <stdio.h>
#define MAX 1000000
static char comp[MAX + 1];              /* comp[i] = 1 মানে composite */

int main(void){
    int n;
    scanf("%d", &n);                    /* ধরে নিই 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)। i×i থেকে শুরু, কারণ তার ছোট গুণিতকগুলো আগের prime-ই কেটে দিয়েছে।

P3. GCD ও LCM BB AP 2023

Given two integers A and B, write a program to compute LCM of A and B. সম্পর্ক: **LCM × GCD = A × B**।

```c
#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;   /* আগে ভাগ, তারপর গুণ — overflow এড়ায় */
    printf("GCD = %d, LCM = %lld\n", g, lcm);
    return 0;
}
```

ধাপ (12, 18)	a	b	r = a % b
১	12	18	12
২	18	12	6
৩	12	6	0
শেষ	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;                /* 20! পর্যন্ত ধরে */
for (int i = 2; i <= n; i++) fact *= i;
printf("%d! = %llu\n", n, fact);
```

Input 5 → 5! = 120 · O(n)। Recursive সংস্করণ P36-এ।

P5. Fibonacci-এর প্রথম n পদ

```
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 সংখ্যা (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++;          /* অঙ্ক সংখ্যা */
    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); */
```

১–১০০০: 1 2 3 4 5 6 7 8 9 153 370 371 407 · $O(d^2)$ প্রতি সংখ্যায় (d = অঙ্ক)

P7. Perfect সংখ্যা (proper divisor-এর যোগফল = সংখ্যা)

```
int isPerfect(int n){
    if (n < 2) return 0;
    int sum = 1;                                  /* 1 সবসময় divisor */
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) {
            sum += i;
            if (i != n / i) sum += n / i;         /* জোড়ার অপর divisor */
        }
    return sum == n;
}
```

6 = 1+2+3, 28 = 1+2+4+7+14, 496 → Perfect · $O(\sqrt{n})$

P8. সংখ্যা উল্টানো, palindrome সংখ্যা ও অঙ্কের যোগফল (একসাথে)

```
int n, t, rev = 0, sum = 0;
scanf("%d", &n);
for (t = n; t > 0; t /= 10) {
    int d = t % 10;          /* শেষ অঙ্ক */
    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: বড় bit আগে ছাপে */
    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)

হাতে রূপান্তর (2020-এর প্রশ্ন)

(i) $(10010.101)_2$ = 16 + 2 + 0.5 + 0.125 = **$(18.625)_{10}$**

(ii) $(543)_{10}$ → 543 ÷ 16 = 33 ভাগশেষ 15 (F); 33 ÷ 16 = 2 ভাগশেষ 1; 2 ÷ 16 = 0 ভাগশেষ 2 → নিচ থেকে উপরে: **$(21F)_{16}$**। যাচাই: 2×256 + 1×16 + 15 = 543 ✔

P10. Power x^n — দ্রুত পদ্ধতি (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;                  /* বিজোড় bit হলে গুণ */
        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 (100 দিয়ে বিভাজ্য কিন্তু 400 নয়), 2000 → Leap

P12. Temporary variable ছাড়া swap

```
a = a + b;  b = a - b;  a = a - b;      /* overflow-এর ঝুঁকি আছে */
/* বা */
a ^= b;     b ^= a;     a ^= b;         /* XOR: a ও b একই মেমরি (একই variable) হলে দুটোই 0
হয়ে যায় */
```

a = 5, b = 9 → a = 9, b = 5

P13. Weekly pay BSCS Officer IT 2020

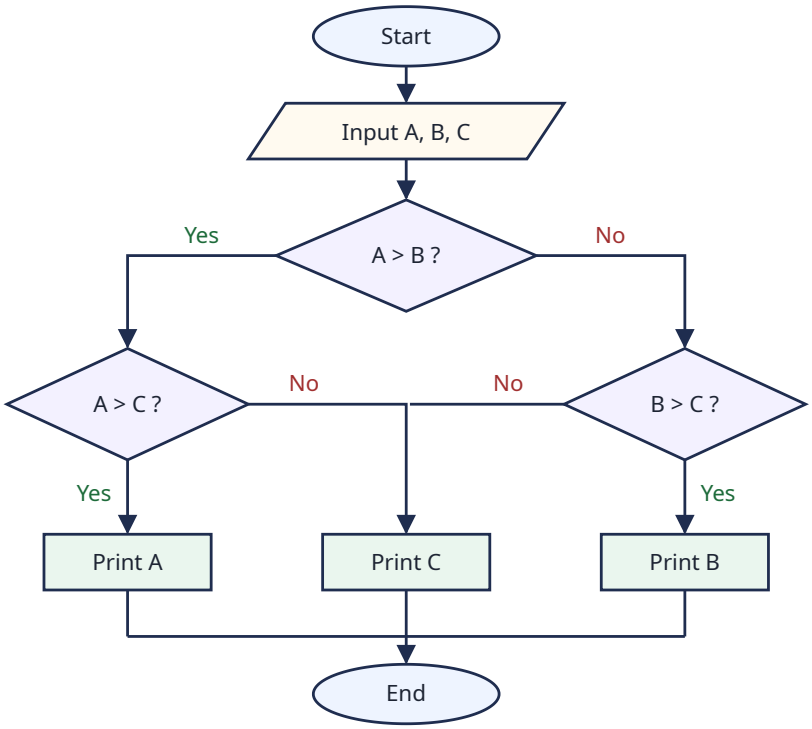
Total weekly pay = hourly wage × regular hours + overtime hours × 1.5 × hourly wage।

```c
#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
# 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. তিনটি সংখ্যার বৃহত্তম — flowchart ও কোড BSCS AP 17.01.2026



চিত্র ৩.১: তিনটি ভিন্ন সংখ্যার বৃহত্তম নির্ণয়ের flowchart

```
if (a > b) { if (a > c) max = a; else max = c; }
else       { if (b > c) max = b; else max = c; }
/* সংক্ষেপে: max = (a > b) ? (a > c ? a : c) : (b > c ? b : c); */
```

Input: 12 45 7 → 45 · তুলনা সবসময় ঠিক ২টি

৩.৩ সিরিজের যোগফল

P15. 1 + 2 + 4 + 7 + 11 + … n-তম পদ পর্যন্ত AP 2024 (10198)

প্যাটার্ন: পার্থক্য 1, 2, 3, 4, … বাড়ে। k-তম পদ $t_k = t_{k-1} + (k - 1) = 1 + k(k - 1)/2$।

$$S_n = \Sigma\,[1 + k(k - 1)/2] = n + (n - 1)n(n + 1)/6 \quad (n = 5\text{: } 5 + 4{\cdot}5{\cdot}6/6 = 25)$$

```c
#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;                      /* পরের পদ = এই পদ + k */
    }
    printf("%lld\n", sum);
    return 0;
}
```

```java
// 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) (সূত্র দিয়ে O(1))

P16. 1 + 1/2 + 1/3 + … + 1/n (Harmonic)

```c
double sum = 0.0;
for (int i = 1; i <= n; i++) sum += 1.0 / i;    /* 1/i লিখলে 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;
/* যাচাই: n(n+1)(2n+1)/6 */
```

n = 5 → 55

P18. $1 - x^2/2! + x^4/4! - \ldots$ (cos x সিরিজ, n পদ)

প্রতিটি পদ আগের পদ থেকে: $t_{k+1} = -t_k \times x^2 / ((2k-1)(2k))$ — আলাদা factorial লাগে না।

```
#include <stdio.h>
int main(void){
    double x; int n;
    scanf("%lf %d", &x, &n);              /* x radian-এ, n = পদ সংখ্যা */
    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)

৩.৪ প্যাটার্ন

> **প্যাটার্নের সূত্র**
> বাইরের loop = সারি (i), ভেতরের loop = কলাম (j)। প্রতি সারিতে কত space ও কত চিহ্ন — i-এর সূত্রে লিখুন (যেমন space = n − i, star = 2i − 1)।

P19. Floyd's triangle (সংখ্যা)

```
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)

সারি i, কলাম j (১ থেকে) হলে মান = (i + j) জোড় হলে 1, বিজোড় হলে 0।

```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)। যাচাই: i=3 → j=1:4 জোড়→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

সারি i-এর পরের মান: C(i, j+1) = C(i, j) × (i − j) / (j + 1)।

```c
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

```c
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);  /* উপরে */
for (int i = n - 1; i >= 1; i--) row(n,
i);  /* নিচে */
```

```
n = 3:
  *
 ***
*****
 ***
  *
```

৩.৫ অ্যারে ও ম্যাট্রিক্স

P24. Function-এ অ্যারে পাঠিয়ে ক্ষুদ্রতম মান Combined 5 Bank AP 2023

```c
#include <stdio.h>

int findMin(const int a[], int n){      /* অ্যারে pointer হয়ে যায়, তাই n আলাদা পাঠাতে হয় */
    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 অ্যারের সাথেই থাকে
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)। Max-এর জন্য শুধু `<` → `>`।

P25. দ্বিতীয় বৃহত্তম (একবার traverse)

```
#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)। Sort করে বের করলে O(n log n) — অপ্রয়োজনীয়।

P26. অ্যারে উল্টানো (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 ও 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;                                  /* পাওয়া যায়নি */
}

int binarySearch(const int a[], int n, int key){   /* a অবশ্যই sorted */
    int lo = 0, hi = n - 1;
    while (lo <= hi) {
        int mid = lo + (hi - lo) / 2;           /* (lo+hi)/2-এর overflow এড়ায় */
        if (a[mid] == key) return mid;
        else if (a[mid] < key) lo = mid + 1;    /* ডান অর্ধেকে খুঁজি */
        else hi = mid - 1;                      /* বাম অর্ধেকে খুঁজি */
    }
    return -1;
}
```

ধাপ (key = 23)	lo	hi	mid	a[mid]	সিদ্ধান্ত
১	0	9	4	16	16 < 23 → lo = 5
২	5	9	7	56	56 > 23 → hi = 6
৩	5	6	5	23	পাওয়া গেছে → index 5

অ্যারে {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}। Binary O(log n), Linear O(n)। Unsorted অ্যারেতে binary search ব্যর্থ হয়, কারণ "a[mid] < key হলে key ডানে আছে" — এই সিদ্ধান্ত তখন সত্য নয়।

P28. Bubble sort (early-exit flag সহ)

```
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++)        /* শেষের i টি আগেই সাজানো */
            if (a[j] > a[j + 1]) {
                int t = a[j]; a[j] = a[j + 1]; a[j + 1] = t;
                swapped = 1;
            }
        if (!swapped) break;                        /* কোনো swap না হলে sorted → best O(n)
*/
    }
}
```

P29. Selection sort

```
void selectionSort(int a[], int n){
    for (int i = 0; i < n - 1; i++) {
        int m = i;                                  /* বাকি অংশের ক্ষুদ্রতমের index */
        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 — পূর্ণ প্রোগ্রাম 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) {     /* বড়গুলো এক ঘর ডানে সরাও */
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;                      /* key ঠিক জায়গায় বসাও */
    }
}

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 · প্রতিটি pass-এর অবস্থা অধ্যায় ৪-এর sorting টেবিলে।

Sort	Best	Average	Worst	Stable	কখন ভালো
Bubble (flag)	O(n)	$O(n^2)$	$O(n^2)$	হ্যাঁ	শেখানোর জন্য, প্রায়-sorted ছোট ডেটা
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	না	swap সবচেয়ে কম ($\le n-1$)
Insertion	O(n)	$O(n^2)$	$O(n^2)$	হ্যাঁ	ছোট বা প্রায়-sorted ডেটা, online input
Merge	O(n log n)	O(n log n)	O(n log n)	হ্যাঁ	বড় ডেটা, linked list; extra O(n) মেমরি

Bubble কেন বড় ডেটায় অদক্ষ, Merge কেন ভালো BSCS AP 17.01.2026

Bubble sort প্রতিটি pass-এ শুধু পাশাপাশি দুটি উপাদান তুলনা করে একটি উপাদানকে ঠিক জায়গায় পাঠায় — n টি উপাদানের জন্য প্রায় $n^2/2$ তুলনা। 10^6 উপাদানে $\approx 5 \times 10^{11}$ তুলনা। Merge sort অ্যারেকে বারবার অর্ধেক করে ($\log_2 n$ স্তর) এবং প্রতি স্তরে n তুলনায় merge করে — মোট $n \log_2 n \approx 2 \times 10^7$ তুলনা, অর্থাৎ প্রায় ২৫,০০০ গুণ কম।

P31. Sorted অ্যারে থেকে duplicate বাদ

```c
int k = 0;                                   /* k = নতুন দৈর্ঘ্য */
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)। Unsorted হলে আগে sort করুন বা nested loop ($O(n^2)$)।

P32. প্রতিটি উপাদানের frequency

```
int a[] = {2, 3, 2, 5, 3, 2}, n = 6, done[6] = {0};
for (int i = 0; i < n; i++) {
    if (done[i]) continue;                  /* আগেই গোনা হয়েছে */
    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 · $O(n^2)$। মানের range ছোট হলে count array দিয়ে O(n)।

P33. Matrix যোগ, গুণ ও 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];        /* যোগ */
            T[j][i] = A[i][j];                  /* transpose */
            P[i][j] = 0;                        /* গুণ: সারি × কলাম */
            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]] (প্রথম সারি: 1·5 + 2·7 = 19, 1·6 + 2·8 = 22) · Transpose(A) = [[1, 3], [2, 4]]। গুণের শর্ত: A-এর কলাম = B-এর সারি; সময় $O(n^3)$।

৩.৬ স্ট্রিং

P34. String উল্টানো ও palindrome

```
#include <stdio.h>
#include <string.h>
int main(void){
    char s[101];
    scanf("%100s", s);                      /* এক শব্দ; লাইনের জন্য fgets */
    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. Vowel, consonant ও word গণনা

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
int main(void){
    char s[201];
    fgets(s, sizeof s, stdin);               /* space-সহ পুরো লাইন */
    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++; }   /* নতুন শব্দ শুরু */
    }
    printf("Vowels=%d Consonants=%d Words=%d\n", v, c, words);
    return 0;
}
```

Input: Bangladesh Bank → Vowels=4 Consonants=10 Words=2। একাধিক space থাকলেও word গণনা ঠিক থাকে।

P36. Anagram পরীক্ষা (listen ↔ silent)

```
int isAnagram(const char *a, const char *b){   /* শুধু ছোট হাতের a-z ধরে নিই */
    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)`

৩.৭ Recursion

Recursion লেখার দুই অংশ

Base case (কোথায় থামবে) + **Recursive case** (ছোট সমস্যায় নিজেকে ডাকা)। Base case না থাকলে stack overflow। প্রতিটি call stack-এ একটি frame রাখে, তাই space O(depth)।

P37. Recursive factorial — algorithm ও কোড 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; মোট call সংখ্যা 15। Time $O(2^n)$ — একই মান বারবার হিসাব হয়; memoization বা loop (P5) দিলে O(n)।

P39. Recursive power ও 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);             /* উপরের n-1 টি সহায়ক খুঁটিতে */
    printf("Move disk %d from %c to %c\n", n, from, to);
    hanoi(n - 1, via, to, from);             /* সেগুলো লক্ষ্যে */
}
/* 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
```

মোট চাল = $2^n - 1$ (n = 3 → 7) · Time $O(2^n)$

৩.৮ Java OOP প্রোগ্রাম

P41. Class, constructor overloading ও 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। Constructor-এর return type নেই, নাম class-এর নামে; 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 Shape, object আলাদা */
            System.out.printf("%s: %.2f%n", s.name(), s.area());
    }
}
```

Output: Circle: 3.14 / Rectangle: 12.00। কোন `area()` চলবে তা runtime-এ object দেখে ঠিক হয় (dynamic dispatch)।

P43. Interface

```
interface Payable { double getPayment(); }          /* চুক্তি: কী করতে হবে */

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। একটি class একাধিক interface implement করতে পারে (Java-তে multiple inheritance-এর বিকল্প)।

P44. Abstract class

```
abstract class Account {
    protected double balance;
    Account(double balance) { this.balance = balance; }
    abstract double interestRate();                  /* subclass-কে দিতেই হবে */
    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। Abstract class-এর object বানানো যায় না; এতে constructor, field ও concrete method থাকতে পারে — interface থেকে এটাই মূল পার্থক্য।

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");           /* সবসময় চলে */
        }
        try {
            System.out.println(a[2]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Bad index");
        }
    }
}
```

Output: Divide by zero / finally runs / Bad index

বিষয়	Checked exception	Unchecked exception
উদাহরণ	IOException, SQLException	ArithmeticException, NullPointerException
কখন ধরা পড়ে	Compile time-এ handle/declare বাধ্যতামূলক	Runtime-এ; handle বাধ্যতামূলক নয়
Parent	Exception (RuntimeException বাদে)	RuntimeException

`throw` = একটি exception ছোড়া (method-এর ভেতরে); `throws` = method signature-এ ঘোষণা যে এটি exception ছুড়তে পারে।

P46. BankAccount — deposit, withdraw ও custom exception

```
class InsufficientBalanceException extends Exception {     /* checked */
    public InsufficientBalanceException(String msg) { super(msg); }
}

class BankAccount {
    private final String accNo;
    private double balance;                                  /* বাইরে থেকে সরাসরি বদলানো যাবে
না */

    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
```

একটি প্রোগ্রামেই encapsulation (private balance + getter), validation, checked custom exception ও finally দেখানো হলো — ব্যাংকের প্রশ্নে এটি সরাসরি কাজে লাগে।

৩.৯ Output Tracing — ২১টি সমস্যা

> **কৌশল**
> প্রতিটি variable-এর জন্য কাগজে একটি কলাম রাখুন এবং প্রতিটি লাইনের পরে মান লিখুন। Recursion হলে প্রতিটি call-এর নিজের local variable আলাদা — call tree আঁকুন।

T1. Post ও pre-increment

```
int i = 5;
printf("%d ", i++);
printf("%d ", ++i);
printf("%d", i);
```

Output: `5 7 7` — i++ আগে মান (5) দেয় পরে বাড়ায় (6); ++i আগে বাড়ায় (7) পরে দেয়।

T2. Integer division ও 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 (ভগ্নাংশ বাদ), float-এ রাখলেও আগেই 3 হয়ে গেছে; C99-এ ভাগ শূন্যের দিকে কাটে, ভাগশেষের চিহ্ন ভাজ্যের মতো।

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'; ছোট হাতের অক্ষর ৩২ বেশি (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` — static একবারই initialize হয় এবং call-এর মাঝে মান ধরে রাখে; সাধারণ local প্রতিবার নতুন।

T5. Recursion — print আগে ও পরে 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	প্রথম printf	পরের call-এ যায়	ফিরে এসে i	দ্বিতীয় 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,` — দ্বিতীয় printf-গুলো উল্টো ক্রমে (ভেতরের call আগে শেষ হয়)।

> **প্রশ্নপত্রের টাইপো**
>
> মূল প্রশ্নে `int fun`, `if(I<=Ø)` ও `Printf` ছাপা ছিল। C case-sensitive, তাই হুবহু লিখলে compile error হতো — উত্তরের শুরুতে "ধরে নিচ্ছি I = i, Printf = printf, Ø = 0" লিখে দিন। `int` function-এ মানহীন `return;` C-তে warning দেয়; output একই।

T6. Pointer ও অ্যারে

```
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 = মান+1; *(p+1) = পরের উপাদান; p[-1] = *(p−1); *++p আগে pointer বাড়ায়।

T7. Call by value বনাম 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 কপি বদলায়; s2 address দিয়ে আসল মান বদলায়।

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-তে break নেই, তাই break পর্যন্ত নিচের case-ও চলে।

T9. Short-circuit evaluation

```
int i = 0, j = 5;
if (i && ++j) ;
printf("%d ", j);
if (i || ++j) ;
printf("%d", j);
```

Output: `5 6` — &&-এর বাম দিক false হলে ডান দিক চলে না; ||-এর বাম দিক false বলে ডান দিক (++j) চলে।

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 ও 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` — জোড় সংখ্যা skip; i = 9-এ break।

T12. while (n--)

```
int n = 3;
while (n--) printf("%d ", n);
printf("| %d", n);
```

Output: `2 1 0 | -1` — শর্তে পুরোনো মান পরীক্ষা হয়, তারপর কমে; শেষ পরীক্ষায় 0 (false) দেখে n হয় −1।

T13. Recursion — call-এর আগে ও পরে print

```
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 গণনা

```
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 == বনাম 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` — literal একই String pool object; `new` নতুন object; == reference তুলনা করে, equals() 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` — compiler আগে নিকটতম widening (byte→int, float→double, char→int) খোঁজে; boxing (Integer) শুধু তা ব্যর্থ হলে।

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` — static method reference type (A) দেখে compile time-এ ঠিক হয়; instance method object (B) দেখে runtime-এ।

T18. Java: field hiding বনাম 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` — field-এ polymorphism নেই (reference type P-এর x); method override হয়েছে।

T19. Java: constructor চলার ক্রম

```
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` — প্রতিটি constructor প্রথমে (অদৃশ্যভাবে হলেও) super() ডাকে, তাই parent আগে।

T20. Java: compound assignment ও increment

```
int x = 5;
x += x++ + ++x;
System.out.println(x);
```

Output: `17` — Java-তে বাম থেকে ডানে: পুরোনো x = 5 সংরক্ষিত; x++ দেয় 5 (x = 6); ++x দেয় 7; 5 + (5 + 7) = 17।

> **C-তে একই কোড = Undefined Behaviour**
>
> C-তে একই expression-এ একটি variable একাধিকবার বদলালে (sequence point ছাড়া) ফল compiler-ভেদে আলাদা — পরীক্ষায় এমন প্রশ্ন এলে "undefined behaviour" লিখে কারণ দিন।

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` — বাম থেকে ডানে; String আসার আগে সংখ্যা যোগ (1+2 = 3), পরে সব concatenation; char + char = int (97 + 98)।

অধ্যায় ৪ — SQL ও ডেটা স্ট্রাকচার হাতে-কলমে

SQL Query Bank & DSA Hand-Simulation

AP লিখিত পরীক্ষায় প্রতিবার অন্তত একটি SQL প্রশ্ন (romantic comedy, S-R-Q join, Department/Employee, IT বিভাগের কর্মী) এবং একটি "হাতে করে দেখাও" DSA প্রশ্ন (max-heap insert/extract, hashing with linear probing, adjacency list/matrix, LRU page fault, queue using two stacks) এসেছে। এই অধ্যায়ের প্রথম ভাগে ৩১টি SQL query (নমুনা ডেটার উপর ফলাফলসহ), দ্বিতীয় ভাগে ধাপে ধাপে DSA টেবিল।

৪.১ নমুনা স্কিমা ও ডেটা

সব query MySQL 8 / standard SQL-এ লেখা। নিচের ডেটার উপর প্রথম ১০টি query-র ফলাফল দেখানো হয়েছে — নিজে মিলিয়ে দেখুন।

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

মোট বেতন 405000, গড় 57857.14। Audit বিভাগে কেউ নেই; Rupa-র বিভাগ NULL — দুটোই ইচ্ছা করে রাখা, যাতে LEFT JOIN ও NULL-এর ফাঁদ দেখানো যায়।

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

৪.২ SQL Query Bank (৩১টি)

Q1. শর্ত ও সাজানো — বেতন ৫৫,০০০-এর বেশি, বেশি থেকে কম

```
SELECT name, salary
FROM employee
WHERE salary > 55000
ORDER BY salary DESC, name ASC;      -- সমান বেতনে নামের ক্রম, যাতে ফল নির্দিষ্ট থাকে
```

name	salary
Rahim	80000
Salma	70000
Karim	60000
Tanvir	60000

Q2. LIKE — pattern matching

```
SELECT name FROM employee WHERE name LIKE 'R%';     -- R দিয়ে শুরু
SELECT name FROM employee WHERE name LIKE '%a';     -- a দিয়ে শেষ
-- '_a%' = দ্বিতীয় অক্ষর a;  % = যেকোনো সংখ্যক অক্ষর, _ = ঠিক একটি অক্ষর
```

ফল: **'R%'** → Rahim, Rupa · **'%a'** → Salma, Nadia, Rupa · **'_a%'** → Rupa বাদে বাকি ৬ জন।

Q3. IN ও BETWEEN

```
SELECT name, dept_id, salary
FROM employee
WHERE dept_id IN (20, 30)
  AND salary BETWEEN 40000 AND 60000;   -- BETWEEN দুই প্রান্তসহ (inclusive)
```

name	dept_id	salary
Nadia	20	50000
Jamil	30	40000

Q4. NULL handling — মোট আয় = বেতন + কমিশন

```
SELECT name, salary + COALESCE(commission, 0) AS total_income   -- MySQL:
IFNULL(commission, 0)
FROM employee;

SELECT name FROM employee WHERE commission IS NULL;     -- "= NULL" কখনো কাজ করে না
```

name	total_income
Rahim	80000
Karim	65000
Salma	70000
Nadia	53000
Tanvir	60000
Jamil	42000
Rupa	45000

commission IS NULL → Rahim, Salma, Tanvir, Rupa।

NULL-এর ফাঁদ

`salary + commission` লিখলে যাদের commission NULL তাদের মোট আয়ও NULL আসবে — NULL-এর সাথে যেকোনো গাণিতিক ক্রিয়া NULL। Aggregate (SUM, AVG, COUNT(col)) NULL বাদ দিয়ে হিসাব করে, কিন্তু COUNT(*) সব সারি গোনে।

Q5. Aggregate function

```
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 — বিভাগভিত্তিক কর্মী ও মোট বেতন (অন্তত ২ জন)

```
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                -- group-এর উপর শর্ত: HAVING; সারির উপর: WHERE
ORDER BY total_salary DESC;
```

dept_name	emp_count	total_salary	avg_salary
IT	3	200000	66666.67
Finance	2	120000	60000.00

লেখার ক্রম বনাম চলার ক্রম

লেখা: SELECT → FROM → WHERE → GROUP BY → HAVING → ORDER BY → LIMIT। চলে: **FROM/JOIN → WHERE → GROUP BY → HAVING → SELECT → ORDER BY → LIMIT**। তাই WHERE-এ aggregate (COUNT, AVG) লেখা যায় না, আর SELECT-এর alias WHERE-এ ব্যবহার করা যায় না (ORDER BY-তে যায়)।

Q7. INNER JOIN — কর্মীর নাম, বিভাগ ও অবস্থান

```
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 আসেনি (dept_id NULL কোনো সারির সাথে মেলে না); Audit-ও আসেনি (কোনো কর্মী নেই)।

Q8. LEFT JOIN — প্রতিটি বিভাগের কর্মী সংখ্যা (শূন্যসহ)

```
SELECT d.dept_name, COUNT(e.emp_id) AS emp_count    -- COUNT(*) দিলে Audit-এ 1 আসত!
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 — কর্মী ও তার ম্যানেজার

```
SELECT e.name AS employee, m.name AS manager
FROM employee e
JOIN employee m ON e.manager_id = m.emp_id;     -- একই টেবিল দুই alias-এ
```

employee	manager
Karim	Rahim
Salma	Rahim
Nadia	Salma
Tanvir	Karim
Jamil	Rahim
Rupa	Rahim

LEFT JOIN দিলে Rahim → NULL সারিটিও আসবে (তার কোনো ম্যানেজার নেই)।

Q10. গড়ের চেয়ে বেশি বেতন (subquery)

```
SELECT name, salary
FROM employee
WHERE salary > (SELECT AVG(salary) FROM employee);     -- গড় 57857.14
```

name	salary
Rahim	80000
Karim	60000
Salma	70000
Tanvir	60000

Q11. নিজের বিভাগের গড়ের চেয়ে বেশি (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);   -- বাইরের প্রতিটি সারির জন্য আলাদা গড়
```

বিভাগের গড়: IT 66666.67, Finance 60000, HR 40000 → ফল: **Rahim (80000), Salma (70000)**। Jamil-এর বেতন গড়ের সমান (বেশি নয়); Rupa-র dept NULL বলে subquery-র গড় NULL → তুলনা unknown → বাদ।

Q12. N-তম সর্বোচ্চ বেতন — চার পদ্ধতি

```
-- (a) দ্বিতীয় সর্বোচ্চ: MAX-এর চেয়ে ছোটদের মধ্যে MAX
SELECT MAX(salary) FROM employee
WHERE salary < (SELECT MAX(salary) FROM employee);              -- 70000

-- (b) LIMIT/OFFSET: N-তম = OFFSET N-1 (DISTINCT জরুরি)
SELECT DISTINCT salary FROM employee
ORDER BY salary DESC LIMIT 1 OFFSET 1;                            -- 70000

-- (c) Window function: তৃতীয় সর্বোচ্চ বেতনের সব কর্মী
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: N-তম হলে তার চেয়ে বড় ভিন্ন বেতন ঠিক N-1 টি
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. Duplicate খোঁজা ও মোছা

```
SELECT salary, COUNT(*) AS cnt
FROM employee
GROUP BY salary
HAVING COUNT(*) > 1;                          -- 60000 → 2

-- একই email-এর ডুপ্লিকেট সারি মুছে সবচেয়ে ছোট id রাখা (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 — এবং NOT IN-এর NULL ফাঁদ

```
-- কর্মীহীন বিভাগ
SELECT d.dept_name FROM department d
WHERE NOT EXISTS (SELECT 1 FROM employee e WHERE e.dept_id = d.dept_id);   -- Audit

-- একই কাজ NOT IN দিয়ে — কিন্তু ফল শূন্য!
SELECT dept_name FROM department
WHERE dept_id NOT IN (SELECT dept_id FROM employee);                       -- কোনো সারি নেই

-- সংশোধন
SELECT dept_name FROM department
WHERE dept_id NOT IN (SELECT dept_id FROM employee WHERE dept_id IS NOT NULL);  -- Audit
```

কেন NOT IN ব্যর্থ হলো

Subquery-তে Rupa-র NULL আছে। `40 NOT IN (10, 20, 10, …, NULL)` মানে `40 <> 10 AND … AND 40 <> NULL`; শেষ অংশ UNKNOWN, তাই পুরো শর্ত কখনো TRUE হয় না। NOT EXISTS-এ এই সমস্যা নেই — nullable কলামে NOT EXISTS ব্যবহার করুন।

Q15. গ্রাহকভিত্তিক মোট ব্যালান্স (১ লাখের বেশি)

```
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;
```

ফল: Arif 170000, Chayan 305000 (Bina 80000 বাদ)।

Q16. শাখাভিত্তিক হিসাব সংখ্যা ও মোট জমা

```
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;
```

ফল: Agrabad 2 টি, 380000 · Motijheel 2 টি, 155000 · Gulshan 1 টি, 20000।

Q17. যাদের কোনো হিসাব নেই

```
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
```

ফল: Dola।

Q18. যে হিসাবে কোনো লেনদেন হয়নি

```
SELECT a.acc_no
FROM account a
WHERE NOT EXISTS (SELECT 1 FROM transactions t WHERE t.acc_no = a.acc_no);
```

ফল: 5002, 5004।

Q19. মাসভিত্তিক জমা ও উত্তোলন (CASE দিয়ে pivot)

```
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: যে সিনেমা Romance *এবং* Comedy দুটোই BB AP 2023

স্কিমা: MOVIE(movie_id, title), GENRE(genre_id, genre), HAS_GENRE(movie_id, genre_id) — junction table।
নমুনা: Movie A → Romance, Comedy; Movie B → Romance; Movie C → Comedy, Action।

```
-- পদ্ধতি ১: junction table দুবার join (একবার Romance, একবার 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';

-- পদ্ধতি ২: 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;
```

ফল: **1** (শুধু Movie A)।

> **সবচেয়ে সাধারণ ভুল**
>
> `WHERE genre = 'Romance' AND genre = 'Comedy'` — একটি সারিতে genre একসাথে দুটো হতে পারে না, তাই ফল সবসময় 0। আবার `OR` দিলে যেকোনো একটি থাকলেই গোনা হবে (ফল 3) — দুটোই ভুল।

Q21. S–R–Q–U: সম্পর্কিত (sid, uid) জোড়া

Combined 5 Bank AP 2023

S(sid, A), T(tid, B), U(uid, C), R(sid, tid, D) — S ও T-এর many-to-many, Q(tid, uid, E) — T ও U-এর many-to-many। SELECT DISTINCT ছাড়া (sid, uid) ফেরত দিতে হবে।

```
SELECT R.sid, Q.uid
FROM R
JOIN Q ON R.tid = Q.tid;          -- একই T-record-এর মাধ্যমে সম্পর্কিত

-- S ও U-তে রেকর্ডের অস্তিত্ব নিশ্চিত করতে চাইলে (FK থাকলে ফল একই)
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;
```

একই (sid, uid) জোড়া একাধিক T-এর মাধ্যমে যুক্ত হলে একাধিকবার আসবে — প্রশ্নে DISTINCT নিষেধ, তাই এটাই প্রত্যাশিত। উত্তরে এক লাইনে এটি উল্লেখ করুন।

Q22. IT বিভাগের কর্মীদের নাম — logical steps

BSCS AP 03.01.2026

```
-- ধাপ ১: DEPARTMENTS থেকে DeptName = 'IT' সারিটি খুঁজি (তার Dept মান)
-- ধাপ ২: EMPLOYEES-কে DeptID = Dept শর্তে join করি
-- ধাপ ৩: শুধু Name কলাম নিই
SELECT e.Name
FROM EMPLOYEES e
JOIN DEPARTMENTS d ON e.DeptID = d.Dept
WHERE d.DeptName = 'IT';

-- subquery বিকল্প
SELECT Name FROM EMPLOYEES
WHERE DeptID = (SELECT Dept FROM DEPARTMENTS WHERE DeptName = 'IT');
```

Q23. Department ও Employee টেবিল (a, b, c) AP 2024 (10198)

```
-- (a) প্রত্যেকের মোট বেতন = salary + commission
SELECT EmpFname, Salary + COALESCE(Commission, 0) AS TotalSalary
FROM Employee;

-- (b) গড়ের চেয়ে বেশি বেতন যাদের — নাম ও অবস্থান, নাম অনুযায়ী সাজানো
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) নামভিত্তিক কর্মী সংখ্যা, সংখ্যা অনুযায়ী ছোট থেকে বড়
SELECT EmpFname, COUNT(EmpCode) AS EmpCount
FROM Employee
GROUP BY EmpFname
ORDER BY EmpCount ASC;
```

লক্ষ করুন: প্রশ্নের CREATE-এ Department.DeptCode `int` কিন্তু Employee.DeptCode `varchar` — join-এ type mismatch; উত্তরে "দুটোই INT হওয়া উচিত, Employee.DeptCode Department-এর FOREIGN KEY" লিখলে বাড়তি নম্বর। (c)-এর ভাষা অস্পষ্ট; নিজের ব্যাখ্যা এক লাইনে লিখে দিন।

Q24. CASE — বেতন গ্রেড

```
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 প্রথম সত্য শর্তেই থামে)।

Q25. UPDATE ও DELETE

```
UPDATE employee SET salary = salary * 1.05 WHERE dept_id = 10;     -- IT-তে ৫% বৃদ্ধি

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';            -- WHERE ছাড়া দিলে সব মুছে
যাবে!
```

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 নিষেধ
    open_date  DATE DEFAULT (CURRENT_DATE),                               -- MySQL 8-এ
বন্ধনী লাগে
    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
);
```

টাকার জন্য FLOAT নয়, DECIMAL — floating-point rounding এড়াতে। CHECK MySQL 8.0.16+ থেকে কার্যকর।

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
```

View = সংরক্ষিত query (virtual table)। সুবিধা: জটিল query সরল করা, নির্দিষ্ট কলাম লুকিয়ে নিরাপত্তা (teller শুধু view দেখবে, মূল টেবিল নয়)।

Q29. Transaction — ফান্ড ট্রান্সফার (Atomicity)

```
START TRANSACTION;

UPDATE account SET balance = balance - 10000
WHERE acc_no = 5001 AND balance >= 10000;     -- affected rows 1 না হলে → 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;          -- সব সফল হলে স্থায়ী;  যেকোনো ধাপে error হলে: ROLLBACK;
```

পরে: 5001 = 140000, 5003 = 90000। মাঝপথে বিদ্যুৎ গেলে বা দ্বিতীয় UPDATE ব্যর্থ হলে ROLLBACK প্রথমটিকেও ফিরিয়ে দেয় — টাকা "হারায়" না। `SAVEPOINT s1;` ... `ROLLBACK TO s1;` দিয়ে আংশিক ফেরানো যায়।

Q30. INDEX

```
CREATE INDEX idx_txn_acc_date ON transactions (acc_no, txn_date);
-- WHERE acc_no = 5001 AND txn_date BETWEEN ... — এই query full scan না করে index ব্যবহার করবে
DROP INDEX idx_txn_acc_date ON transactions;
```

Index (সাধারণত B+ tree) খোঁজা O(n) থেকে প্রায় O(log n) করে, কিন্তু INSERT/UPDATE ধীর করে ও জায়গা নেয়। Composite index-এ বাম দিকের কলাম (acc_no) ছাড়া শুধু txn_date দিয়ে খুঁজলে সাধারণত কাজে লাগে না।

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'@'%';    -- শুধু view
REVOKE INSERT ON bankdb.transactions FROM 'teller'@'%';
```

Least privilege: প্রত্যেক ব্যবহারকারী শুধু কাজের জন্য যতটুকু দরকার ততটুকু অনুমতি পাবে।

৪.৩ Max-Heap: insert ও extract-root Combined 5 Bank AP 2023

প্রশ্ন: Insert 5, 6, 8, Extract, Insert 4, 11, Extract, Insert 7, Extract, Extract — প্রতিটির পর heap দেখাও এবং extract-এ কোন মান ফেরত আসে বলো।

নিয়ম (array-ভিত্তিক, index 0 থেকে)

parent(i) = (i − 1)/2, left = 2i + 1, right = 2i + 2।

Insert: শেষে বসাও → parent-এর চেয়ে বড় হলে উপরে swap (sift-up)।

Extract: root ফেরত দাও → শেষ উপাদান root-এ আনো → বড় child-এর সাথে নিচে swap (sift-down/heapify)।

দুটোই O(log n)।

ধাপ	অপারেশন	কী হলো	Heap (array)	ফেরত
a	Insert 5	খালি heap-এ root	[5]	—
b	Insert 6	6 > 5 → swap	[6, 5]	—
c	Insert 8	index 2-এ, parent 6 < 8 → swap	[8, 5, 6]	—
d	Extract	8 সরাও, শেষের 6 root-এ; child 5 < 6 → ঠিক আছে	[6, 5]	**8**
e	Insert 4	parent 6 > 4 → ঠিক আছে	[6, 5, 4]	—
f	Insert 11	index 3, parent 5 → swap; parent 6 → swap	[11, 6, 4, 5]	—
g	Extract	11 সরাও, 5 root-এ; বড় child 6 → swap	[6, 5, 4]	**11**
h	Insert 7	index 3, parent 5 → swap; parent 6 → swap	[7, 6, 4, 5]	—
i	Extract	7 সরাও, 5 root-এ; বড় child 6 → swap	[6, 5, 4]	**7**
j	Extract	6 সরাও, 4 root-এ; child 5 > 4 → swap	[5, 4]	**6**

Extract-এ ফেরত আসা মানের ক্রম: **8, 11, 7, 6**; শেষে heap = [5, 4]। পরীক্ষায় প্রতিটি ধাপের পাশে ছোট tree এঁকে দিন।

8.8 Min-Heap তৈরি (bottom-up heapify)

Array [9, 4, 7, 1, 8, 2, 3] থেকে min-heap। শেষ non-leaf index = ⌊n/2⌋ − 1 = 2 থেকে 0 পর্যন্ত sift-down।

i	উপাদান	children	কাজ	Array
2	7	2, 3	ছোট child 2-এর সাথে swap	[9, 4, 2, 1, 8, 7, 3]
1	4	1, 8	1-এর সাথে swap	[9, 1, 2, 4, 8, 7, 3]
0	9	1, 2	1-এর সাথে swap; তারপর index 1-এ 9-এর children 4, 8 → 4-এর সাথে swap	[1, 4, 2, 9, 8, 7, 3]

যাচাই: 1 ≤ (4, 2); 4 ≤ (9, 8); 2 ≤ (7, 3) ✔ · Bottom-up build O(n), একে একে insert করলে O(n log n)।

8.৫ Hashing — চার পদ্ধতিতে একই key সেট

Table size m = 11; keys ক্রমে: 10, 22, 31, 4, 15, 28, 17, 88, 59; h(k) = k mod 11 → 10, 0, 9, 4, 4, 6, 6, 0, 4।

(ক) Linear probing: h(k, i) = (h(k) + i) mod 11

Key	h(k)	Probe ক্রম	বসল
10, 22, 31, 4, 28	10, 0, 9, 4, 6	খালি	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 (৫ probe)

(খ) Quadratic probing: h(k, i) = (h(k) + i^2) mod 11

10, 22, 31, 4, 28 একই জায়গায়; 15: 4 ✗ → 4+1 = 5 · 17: 6 ✗ → 7 · 88: 0 ✗ → 1 · 59: 4 ✗ → 5 ✗ → 4+4 = 8 → **8 (৩ probe)**। এই সেটে চূড়ান্ত টেবিল linear-এর মতোই, কিন্তু 59-এর probe কম — primary clustering (একটানা ভরা ঘরের জট) এড়ানোই quadratic-এর উদ্দেশ্য।

(গ) 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 ক্রম	বসল
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

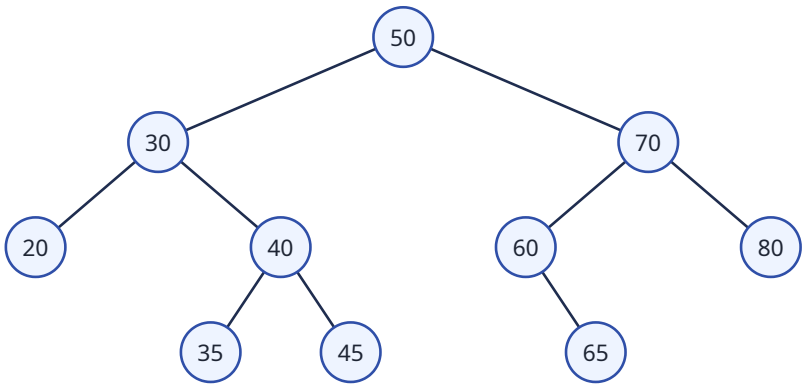
(ঘ) Separate chaining (প্রতিটি ঘরে linked list, নতুনটি শেষে)

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। Open addressing-এ α < 1 হতেই হবে এবং α বাড়লে probe দ্রুত বাড়ে; chaining-এ α > 1-ও চলে, গড় খোঁজা O(1 + α)। Double hashing-এ h_2 কখনো 0 হওয়া চলবে না এবং m prime হলে সব ঘরে পৌঁছানো নিশ্চিত।

৪.৬ Binary Search Tree — insert, traversal, delete

Insert ক্রম: 50, 30, 70, 20, 40, 60, 80, 35, 45, 65 (ছোট হলে বামে, বড় হলে ডানে)।



চিত্র ৪.১: 50, 30, 70, 20, 40, 60, 80, 35, 45, 65 insert করার পর BST

Traversal	নিয়ম	ফল
Inorder	Left → Root → Right	20 30 35 40 45 50 60 65 70 80 (সবসময় 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	স্তর ধরে (queue)	50 30 70 20 40 60 80 35 45 65

Delete-এর তিন ক্ষেত্র (প্রতিটি মূল tree থেকে আলাদাভাবে)

Delete	ধরন	কী করবেন	ফল
20	Leaf	সরাসরি মুছে ফেলুন	30-এর বাম child খালি
60	এক child (65)	child-কে parent-এর সাথে জুড়ে দিন	70-এর বাম child হলো 65
30	দুই child	Inorder successor (ডান subtree-র ক্ষুদ্রতম = 35) দিয়ে মান বদলান, তারপর 35-কে মুছুন	30-এর জায়গায় 35; তার children 20 ও 40; 40-এর শুধু ডানে 45

Search/insert/delete: balanced হলে O(log n), কিন্তু sorted ক্রমে insert করলে tree linked list হয়ে যায় → O(n); এ জন্যই AVL/Red-Black।

৪.৭ AVL rotation

Balance factor (BF) = বাম উচ্চতা − ডান উচ্চতা; কোনো node-এ |BF| > 1 হলে rotation।

ক্ষেত্র	Insert ক্রম	সমস্যা	সমাধান	ফল
LL	30, 20, 10	30-এ BF = +2, নতুন node বামের বামে	30-এ right rotation	20 (10, 30)
RR	10, 20, 30	10-এ BF = −2, ডানের ডানে	10-এ left rotation	20 (10, 30)
LR	30, 10, 20	30-এ BF = +2, বামের ডানে	10-এ left, তারপর 30-এ right	20 (10, 30)
RL	10, 30, 20	10-এ BF = −2, ডানের বামে	30-এ right, তারপর 10-এ left	20 (10, 30)

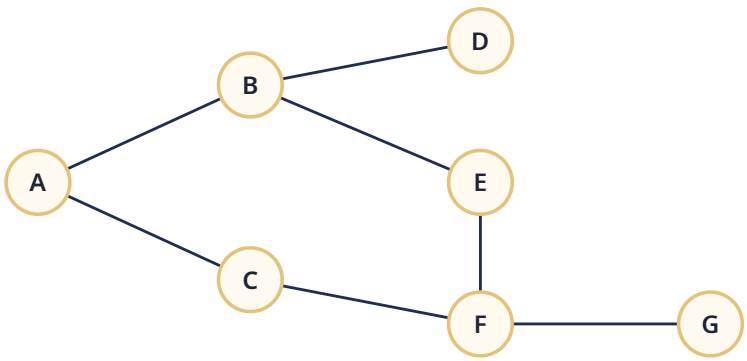
উদাহরণ: 10, 20, 30, 40, 50, 25 insert

10, 20, 30 → RR → 20(10, 30)। 40 → 30-এর ডানে। 50 → 30-এ BF = −2 (RR) → 30-এ left rotation → 20(10, 40(30, 50))।

25 → 20 → 40 → 30-এর বামে। এখন 20-এ BF = 1 − 3 = −2; ডান child 40-এর BF = +1 → **RL**: 40-এ right rotation (30 উপরে), তারপর 20-এ left rotation।

চূড়ান্ত: root 30; বামে 20 (10, 25); ডানে 40 (ডানে 50)। সব node-এ |BF| ≤ 1 ✔

৪.৮ Graph traversal: BFS ও DFS



চিত্র ৪.২: Undirected graph — A: B, C · B: A, D, E · C: A, F · D: B · E: B, F · F: C, E, G · G: F

BFS (A থেকে, queue)		
Dequeue	**নতুন 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 আগেই)	F
F	G	G
G	—	খালি

DFS (A থেকে, recursion/stack)	
পরিদর্শন	**কেন**
A	শুরু
B	A-র প্রথম প্রতিবেশী
D	B-র প্রথম অদেখা
E	D শেষ → B-তে ফিরে E
F	E-র অদেখা প্রতিবেশী
C	F-এর প্রথম অদেখা
G	C শেষ → F-এ ফিরে G

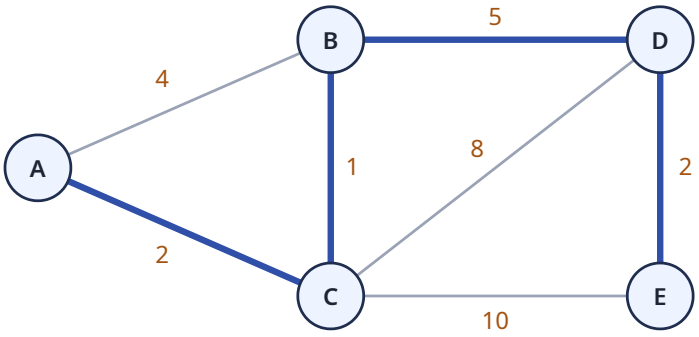
BFS ক্রম: A B C D E F G (স্তর: A=0; B, C=1; D, E, F=2; G=3 — unweighted graph-এ shortest path দেয়) · **DFS ক্রম:** A B D E F C G · দুটোই O(V + E)।

Adjacency list বনাম matrix — কোনটিতে কী দ্রুত Combined 5 Bank AP 2023

Matrix: "u ও v-এর মধ্যে edge আছে?" — O(1); dense graph-এর জন্য ভালো; জায়গা $O(V^2)$।

List: "u-এর সব প্রতিবেশী" বা BFS/DFS — O(deg u) ও মোট O(V + E); sparse graph-এর জন্য ভালো; জায়গা O(V + E)।

৪.৯ Dijkstra — shortest path



চিত্র ৪.৩: Weighted undirected graph (মোটা নীল = MST, ৪.১০ দেখুন)

ধাপ	বেছে নেওয়া (finalized)	A	B	C	D	E
০	—	0	∞	∞	∞	∞
১	A (0)	0	4 (A)	2 (A)	∞	∞
২	C (2)	0	**3 (C)**	2	10 (C)	12 (C)
৩	B (3)	0	3	2	**8 (B)**	12
৪	D (8)	0	3	2	8	**10 (D)**

৫	E (10)	0	3	2	8	10

ফল: A→C = 2; A→B = 3 (A–C–B); A→D = 8 (A–C–B–D); A→E = 10 (A–C–B–D–E)। বন্ধনীতে predecessor। Binary heap দিয়ে O((V + E) log V); negative weight থাকলে Dijkstra চলে না (Bellman-Ford লাগে)।

৪.১০ Minimum Spanning Tree — Kruskal ও Prim (চিত্র ৪.৩)

Kruskal (ছোট edge আগে, cycle হলে বাদ)

Edge	Weight	সিদ্ধান্ত
B–C	1	নিই
A–C	2	নিই
D–E	2	নিই
A–B	4	বাদ (A–C–B cycle)
B–D	5	নিই → ৪টি edge, শেষ

Prim (A থেকে, সবচেয়ে সস্তা বাইরের edge)

Tree-তে আছে	নেওয়া edge	Weight
{A}	A–C	2
{A, C}	C–B	1
{A, B, C}	B–D	5
{A, B, C, D}	D–E	2

দুই পদ্ধতিতেই MST = {B–C, A–C, D–E, B–D}, মোট ওজন **10**; V − 1 = 4টি edge। Kruskal O(E log E) (sort + union-find), Prim O(E log V) (heap)।

৪.১১ 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।

ধাপ	Queue থেকে বের	In-degree আপডেট	Queue
শুরু	—	in-degree 0: A, B	A B
১	A	C: 2→1	B
২	B	C: 1→0, D: 1→0	C D
৩	C	E: 2→1	D
৪	D	E: 1→0	E
৫	E	F: 1→0	F
৬	F	—	খালি

ক্রম: **A B C D E F** (B A D C E F-ও বৈধ — topological order একাধিক হতে পারে)। সব node বের না হলে graph-এ cycle আছে। O(V + E)। ব্যবহার: course prerequisite, build dependency, task scheduling।

৪.১২ Sorting — প্রতিটি pass, একই অ্যারে [29, 10, 14, 37, 13]

Algorithm	Pass ১	Pass ২	Pass ৩	Pass ৪
Bubble	[10, 14, 29, 13, **37**]	[10, 14, 13, **29, 37**]	[10, 13, **14, 29, 37**]	কোনো swap নেই → থামো

Selection	min 10 ↔ 29: [**10**, 29, 14, 37, 13]	min 13 ↔ 29: [10, **13**, 14, 37, 29]	min 14 (নিজ স্থানে)	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: পরিবর্তন নেই	key 13: [10, 13, 14, 29, 37]

Bubble-এ মোট swap 5 = অ্যারের inversion সংখ্যা ((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 = শেষ উপাদান)

```
[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)
ডান [14, 37, 29], pivot 29:
 14 ≤ 29 → নিজ স্থানে; 37 > 29
 swap → [14, 29, 37]
ফল: [10, 13, 14, 29, 37]
```

Quick sort-এর worst case $O(n^2)$ কখন

অ্যারে আগেই sorted (বা উল্টো sorted) এবং pivot সবসময় শেষ/প্রথম উপাদান — তখন প্রতিবার partition (n − 1) ও 0-তে ভাগ হয়, গভীরতা n, মোট তুলনা n(n − 1)/2। সমাধান: random pivot বা median-of-three।

৪.১৩ Binary search trace

Sorted অ্যারে (index 0–9): [3, 8, 15, 21, 27, 34, 42, 56, 63, 77]; mid = ⌊(lo + hi)/2⌋।

key = 63 (আছে)

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 (নেই)

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

সর্বোচ্চ তুলনা $\lceil \log_2(n+1) \rceil$: n = 10 → 4; n = 10^6 → 20।

৪.১৪ Infix → Postfix ও Prefix, এবং Postfix মূল্যায়ন

Expression: **(A + B) * C − D / E**। Precedence: ^ > * / > + −; সমান precedence-এ বাম থেকে ডানে (^ ছাড়া)।

Token	Stack	Output	মন্তব্য
(	(		push
A	(	A	operand → output

+	(+	A	
B	(+	A B	
)		A B +	(পর্যন্ত pop
*	*	A B +	
C	*	A B + C	
−	−	A B + C *	* (বেশি/সমান precedence) pop, তারপর − push
D	−	A B + C * D	
/	− /	A B + C * D	/ বেশি precedence → push
E	− /	A B + C * D E	
শেষ		A B + C * D E / −	stack খালি করো

Postfix: A B + C * D E / − · **Prefix:** − * + A B C / D E (expression উল্টে, বন্ধনী অদলবদল করে postfix বের করে আবার উল্টানো; বা সরাসরি: মূল operator − আগে)।

মূল্যায়ন: 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]**। যাচাই: (2 + 3) × 4 − 10/5 = 20 − 2 = 18 ✔

অতিরিক্ত অনুশীলন: `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**।

নিয়ম: operator এলে দুটি pop — প্রথমটি ডান operand, দ্বিতীয়টি বাম (a − b-তে ক্রম গুরুত্বপূর্ণ)।

৪.১৫ দুটি stack দিয়ে Queue BSCS AP 17.01.2026

লজিক

Enqueue(x): S1-এ push। **Dequeue():** S2 খালি হলে S1-এর সব উপাদান একে একে pop করে S2-তে push (ক্রম উল্টে যায়, সবচেয়ে পুরোনোটি S2-র top-এ); তারপর S2 থেকে pop। S2 খালি না থাকলে সরাসরি S2 থেকে pop। প্রতিটি উপাদান সর্বোচ্চ দুবার সরে → amortized O(1)।

অপারেশন	S1 (নিচ → top)	S2 (নিচ → top)	Output
enqueue 1	1	—	
enqueue 2	1 2	—	
enqueue 3	1 2 3	—	
dequeue	—	3 2 (1 pop হলো)	1
enqueue 4	4	3 2	
dequeue	4	3	2
dequeue	4	—	3
dequeue	—	— (4 সরিয়ে pop)	4

Output 1, 2, 3, 4 — ঢোকার ক্রমেই বের হলো (FIFO) ✔

```
void enqueue(int x){ push(&S1, x); }
int dequeue(void){
    if (isEmpty(&S2))
        while (!isEmpty(&S1)) push(&S2, pop(&S1));   /* উল্টে দাও */
    if (isEmpty(&S2)) { printf("Queue empty\n"); return -1; }
    return pop(&S2);
}
```

8.১৬ 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**, ৩টি frame। F = page fault, — = hit।

Ref	1	2	3	4	2	3	4	1	2	1	1	3	1	4
LRU — সবচেয়ে আগে ব্যবহৃত page বাদ														
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 — সবচেয়ে আগে ঢোকা page বাদ														
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 — যে page সবচেয়ে দেরিতে (বা আর কখনো না) লাগবে তা বাদ														
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 fault: LRU = 8, FIFO = 8, Optimal = 6 (hit ratio যথাক্রমে 6/14, 6/14, 8/14)।

LRU-র কঠিন সিদ্ধান্তগুলো ব্যাখ্যাসহ

Ref 4 (৪র্থ): 1, 2, 3-এর মধ্যে 1 সবচেয়ে আগে ব্যবহৃত → 1 বাদ।

Ref 1 (৮ম): শেষ ব্যবহার — 2 (৫ম), 3 (৬ষ্ঠ), 4 (৭ম) → 2 বাদ।

Ref 2 (৯ম): 3 (৬ষ্ঠ), 4 (৭ম), 1 (৮ম) → 3 বাদ।

Ref 3 (১২তম): 4 (৭ম), 2 (৯ম), 1 (১১তম) → 4 বাদ।

Ref 4 (১৪তম): 2 (৯ম), 3 (১২তম), 1 (১৩তম) → 2 বাদ।

Belady's anomaly

FIFO-তে frame বাড়ালে fault কখনো কখনো *বাড়ে*। Reference 1 2 3 4 1 2 5 1 2 3 4 5: ৩ frame-এ 9 fault, ৪ frame-এ 10 fault। LRU ও Optimal "stack algorithm", তাই এদের এই সমস্যা নেই। Optimal বাস্তবে করা যায় না (ভবিষ্যৎ জানা লাগে) — শুধু তুলনার মানদণ্ড।

৪.১৭ Complexity cheat-sheet

Data structure / Algorithm	Access / Search	Insert	Delete	মন্তব্য
Array	O(1) / O(n)	O(n)	O(n)	শেষে insert O(1) (dynamic array amortized)
Singly linked list	O(n)	শুরুতে O(1)	শুরুতে O(1)	শেষে insert tail pointer থাকলে O(1)
Stack / Queue	—	O(1)	O(1)	LIFO / FIFO
Hash table	গড় O(1), worst O(n)	গড় O(1)	গড় O(1)	সব key একই slot-এ গেলে worst
BST	গড় O(log n), worst O(n)	একই	একই	skewed হলে O(n)
AVL / Red-Black	O(log n)	O(log n)	O(log n)	সবসময় balanced
Binary heap	max/min O(1)	O(log n)	extract O(log n)	build O(n)
Binary search	O(log n)	—	—	sorted array লাগে
BFS / DFS	O(V + E) (list), O(V^2) (matrix)			BFS → unweighted shortest path
Dijkstra	O((V + E) log V) heap দিয়ে			negative weight নয়
Kruskal / Prim	O(E log E) / O(E log V)			MST
Merge / Heap sort	O(n log n) সব ক্ষেত্রে			merge stable, heap in-place
Quick sort	গড় O(n log n), worst O(n^2)			in-place, বাস্তবে দ্রুততম
Bubble / Insertion / Selection	O(n^2) (bubble-flag ও insertion best O(n))			ছোট ডেটা
Tower of Hanoi / naive Fibonacci	O(2^n)			exponential

অধ্যায় ৫ — ১০ নম্বরের থিওরি: রেডি উত্তর

High-Yield Theory Answers for AP Written

BIBM-এর AP পেপারে ১৫০ নম্বরের মধ্যে প্রায় অর্ধেক প্রশ্ন সরাসরি থিওরি — "Explain...", "Differentiate...", "Compare..."। AP ২০২৪ (10198), AP ০৩.০১.২০২৬ ও AP ১৭.০১.২০২৬-এ একই ধরনের প্রশ্ন ঘুরেফিরে এসেছে (Inheritance, Polymorphism, SDLC, Waterfall vs Agile, TCP vs UDP, OSI, Process vs Thread, Hashing vs Encryption, CIA, Cloud models)। এই অধ্যায়ে প্রতিটির **পরীক্ষায় হুবহু লেখার মতো ইংরেজি উত্তর** দেওয়া হলো — সংজ্ঞা, টেবিল, উদাহরণ। প্রতি উত্তর ১০–১২ মিনিটে লেখা যায়।

> **উত্তর লেখার ফর্মুলা**
> Definition (১–২ লাইন) → টেবিল/ডায়াগ্রাম → উদাহরণ (সম্ভব হলে ব্যাংকের) → এক লাইনের উপসংহার।
> "Differentiate" প্রশ্নে কমপক্ষে ৫ সারির টেবিল দিন।

৫.১ Object-Oriented Programming

Q1. What are the four pillars of OOP? AP ১৭.০১.২০২৬ 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

বাংলা টিপ: চারটির নাম মনে রাখুন "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()`

> **ফাঁদ**
> Static method override হয় না — subclass-এ একই signature-এর static method লিখলে তা **method hiding**। আর instance method-কে static দিয়ে (বা উল্টো) প্রতিস্থাপন করলে compile error (BB AP 2023-এর A/B প্রশ্ন)।

Q6. Explain inheritance and its types. AP ১৭.০১.২০২৬ 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

৫.২ 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)

বাংলা টিপ: 2NF = "পুরো key-র উপর নির্ভর", 3NF = "শুধু key-র উপর নির্ভর", BCNF = "বাম দিকে সবসময় 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

*SELECT-কে অনেকে আলাদা DQL (Data Query Language) বলেন।

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 ১৭.০১.২০২৬

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.

বাংলা টিপ: "Relation"-এর প্রকারভেদ জিজ্ঞাসা করলে relationship cardinality (1:1, 1:N, M:N) এবং প্রয়োজনে unary/binary/ternary degree — দুটোই এক লাইনে উল্লেখ করুন।

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.

৫.৬ Operating System

Q1. Process vs Thread; advantages of threads AP ১৭.০১.২০২৬

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 ১৭.০১.২০২৬

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);
}                            }
```

বাংলা টিপ: wait(mutex)-কে wait(empty)-এর আগে দিলে deadlock হতে পারে — এই ক্রমটি উল্লেখ করলে বাড়তি নম্বর।

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 ০৩.০১.২০২৬

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.

৫.৪ Networking

Q1. OSI layers and functions; OSI vs TCP/IP AP ০৩.০১.২০২৬

#	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).

মনে রাখার বাক্য (৭→১): "All People Seem To Need Data Processing"।

Q2. Compare TCP and UDP with examples AP ০৩.০১.২০২৬

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)

উদাহরণ: 192.168.10.0/24-কে ৪টি সাবনেটে

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 ১৭.০১.২০২৬

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

৫.৫ Information Security

Q1. CIA triad AP ০৩.০১.২০২৬ 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 ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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.

৫.৬ Software Engineering

Q1. SDLC and its phases AP ১৭.০১.২০২৬

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 ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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 ১৭.০১.২০২৬

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

বাংলা টিপ: commit = নিজের খাতায় লিখে রাখা; push = সবার সাথে শেয়ার করা। commit অফলাইনেও করা যায়, push-এর জন্য রিমোট লাগে।

৫.৭ Web, Cloud ও AI

Q1. Cloud computing — service and deployment models AP ০৩.০১.২০২৬

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 ০৩.০১.২০২৬

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 ১৭.০১.২০২৬

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 ১৭.০১.২০২৬

- **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.

৫.৮ Digital Logic, Data Communication ও 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 ১৭.০১.২০২৬

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 ০৩.০১.২০২৬

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 ১৭.০১.২০২৬

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.

উদাহরণ (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 ০৩.০১.২০২৬

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.

অধ্যায় ৬ — AP MCQ ব্যাংক (১৫০)

MCQ Bank modelled on BIBM papers (IT 110 · Math 25 · GK 15)

Job ID-25106-এর MCQ অংশে ১০০ নম্বর (Subject ৭৫ + Math ১৫ + GK ১০), ১ ঘণ্টা — প্রশ্নপ্রতি গড়ে ৩৬ সেকেন্ড। নিচের প্রশ্নগুলো BIBM-এর Combined Bank Officer (IT) MCQ ২০২৬ ও বিগত AP পেপারের ধরন অনুসরণ করে বানানো।

> **নেগেটিভ মার্কিং (−০.২৫) কৌশল**
>
> প্রতিটি ভুলে ০.২৫ কাটা। অন্তত **একটি অপশন নিশ্চিতভাবে বাদ** দিতে পারলে আন্দাজ লাভজনক (প্রত্যাশিত নম্বর +০.১৭); দুটি বাদ দিলে +০.৩৭৫। কিছুই বাদ দিতে না পারলে ছেড়ে দিন। প্রথম রাউন্ডে নিশ্চিত প্রশ্নগুলো, দ্বিতীয় রাউন্ডে বাকিগুলো।

> **কীভাবে অনুশীলন করবেন**
>
> প্রতি ৫০টি প্রশ্নে ৩০ মিনিট টাইমার দিন। উত্তর ঢেকে রাখুন, শেষে মিলিয়ে (সঠিক × ১ − ভুল × ০.২৫) হিসাব করুন।

৬.১ Programming ও OOP (১–২০)

1. C-তে `printf("%d", 7 / 2);` এর আউটপুট কী?

(ক) 3.5 (খ) 3.0
(গ) 4 (ঘ) 3

উত্তর: (ঘ) — দুটি int-এর ভাগ integer division, দশমিক অংশ বাদ যায়।

2. Java-তে `System.out.println(5 + 3 + "AB");` এর আউটপুট?

(ক) Compile error (খ) 8AB
(গ) 53AB (ঘ) AB8

উত্তর: (খ) — বাম থেকে মূল্যায়ন: 5 + 3 = 8, তারপর "AB" যুক্ত।

3. Java-তে `System.out.println("AB" + 5 + 3);` এর আউটপুট?

(ক) AB 5 3 (খ) AB53
(গ) 8AB (ঘ) AB8

উত্তর: (খ) — প্রথমেই String, তাই পরের + গুলো concatenation।

4. C-তে `int x = 5; printf("%d", x << 2);` এর আউটপুট?

(ক) 20 (খ) 7
(গ) 25 (ঘ) 10

উত্তর: (ক) — left shift 2 = $\times 2^2 = 5 \times 4 = 20$।

5. নিচের কোডের আউটপুট? `int i; for(i = 0; i < 5; i++); printf("%d", i);`

(ক) 0 1 2 3 4 (খ) 5
(গ) কিছুই না (ঘ) 4

উত্তর: (খ) — for-এর পরে সেমিকোলন, তাই লুপ ফাঁকা; শেষে i = 5, printf একবার চলে।

6. C-তে `int a[5] = {1, 2};` হলে a[3]-এর মান?

(ক) Garbage
(খ) 2
(গ) Error
(ঘ) 0

উত্তর: (ঘ) — আংশিক initialization-এ বাকি element 0 হয়।

7. `int f(int n){ return n == 0 ? 0 : n + f(n - 1); }` হলে f(4) = ?

(ক) 4
(খ) 24
(গ) 10
(ঘ) 6

উত্তর: (গ) — 4 + 3 + 2 + 1 + 0 = 10।

8. Python-এ `print(2 ** 3 ** 2)` এর আউটপুট?

(ক) 64
(খ) 36
(গ) 12
(ঘ) 512

উত্তর: (ঘ) — ** ডান থেকে বামে: $3^2 = 9$, $2^9 = 512$।

9. C-তে `printf("%d", sizeof(char));` এর আউটপুট?

(ক) 8
(খ) 2
(গ) 4
(ঘ) 1

উত্তর: (ঘ) — C standard অনুযায়ী sizeof(char) সবসময় 1।

10. OOP-এর চার স্তম্ভের মধ্যে কোনটি নয়?

(ক) Compilation
(খ) Encapsulation
(গ) Polymorphism
(ঘ) Inheritance

উত্তর: (ক) — চার স্তম্ভ: Abstraction, Encapsulation, Inheritance, Polymorphism।

11. ডেটা ও মেথডকে এক ইউনিটে বেঁধে ডেটা লুকিয়ে রাখাকে কী বলে?

(ক) Abstraction
(খ) Encapsulation
(গ) Modularity
(ঘ) Polymorphism

উত্তর: (খ) — BIBM Officer IT MCQ 2026-এ প্রায় হুবহু এসেছে।

12. Method overloading কখন resolve হয়?

(ক) Link time
(খ) Load time
(গ) Compile time
(ঘ) Run time

উত্তর: (গ) — compile-time (static) polymorphism; overriding হয় run time-এ।

13. Java-তে কোনো class-কে inherit হওয়া থেকে আটকাতে কোন keyword?

(ক) static
(খ) abstract
(গ) private
(ঘ) final

উত্তর: (ঘ) — `final class` extend করা যায় না।

14. Java-তে class-এর multiple inheritance কীভাবে অর্জন করা হয়?

(ক) সম্ভব নয়, কোনো উপায়েই
(খ) Abstract class দিয়ে
(গ) Package দিয়ে
(ঘ) Interface দিয়ে

উত্তর: (ঘ) — একটি class একাধিক interface implement করতে পারে।

15. Constructor-এর return type কী?

(ক) কোনো return type নেই
(খ) class-এর type
(গ) int
(ঘ) void

উত্তর: (ক) — constructor-এ return type লেখা হয় না, void-ও না।

16. Java-তে কোনটি primitive data type নয়?

(ক) char
(খ) boolean
(গ) int
(ঘ) String

উত্তর: (ঘ) — String একটি class (reference type)।

17. Java-তে instance variable `int count;`-এর default মান?

(ক) −1
(খ) 0
(গ) Garbage
(ঘ) null

উত্তর: (খ) — instance/static numeric field-এর default 0 (local variable-এর default নেই)।

18. কোন access modifier শুধু একই class-এর ভেতরে দৃশ্যমান?

(ক) public
(খ) private
(গ) default
(ঘ) protected

উত্তর: (খ)

19. Subclass-এ superclass-এর একই signature-এর static method লিখলে তাকে কী বলে?

(ক) Method hiding
(খ) Overloading
(গ) Overriding
(ঘ) Shadowing error

উত্তর: (ক) — static method override হয় না, hide হয়।

20. C-তে `int x = 10; int y = (x > 5) ? x * 2 : x / 2; printf("%d", y);`

(ক) 10
(খ) 20
(গ) 5
(ঘ) 15

উত্তর: (খ) — শর্ত সত্য, তাই x * 2 = 20।

৬.২ Data Structure ও Algorithm (২১–৪০)

21. Stack কোন নীতিতে চলে?

(ক) Priority
(খ) Random
(গ) FIFO
(ঘ) LIFO

উত্তর: (ঘ) — Last In First Out।

22. Balanced parentheses যাচাই করতে কোন ডেটা স্ট্রাকচার লাগে?

(ক) Graph
(খ) Tree
(গ) Stack
(ঘ) Queue

উত্তর: (গ) — খোলা বন্ধনী push, বন্ধ পেলে pop করে মিলানো।

23. Recursion বাস্তবায়নে সাধারণত কোন ডেটা স্ট্রাকচার ব্যবহৃত হয়?

(ক) Hash table
(খ) Stack
(গ) Queue
(ঘ) Heap

উত্তর: (খ) — function call stack।

24. Sorted array-তে binary search-এর time complexity?

(ক) O(n)
(খ) O(n log n)
(গ) O(log n)
(ঘ) O(1)

উত্তর: (গ) — প্রতি ধাপে search space অর্ধেক।

25. ১০,০০,০০০ ($১০^{৬}$) sorted উপাদানে binary search-এ সর্বোচ্চ প্রায় কতটি তুলনা?

(ক) 20
(খ) 10
(গ) 1000
(ঘ) 500000

উত্তর: (ক) — $2^{20} \approx 10^6$, তাই ≈ 20 তুলনা।

26. Quick sort-এর worst-case time complexity?

(ক) O(n)
(খ) O(n log n)
(গ) O(log n)
(ঘ) $O(n^2)$

উত্তর: (ঘ) — pivot সবসময় সবচেয়ে ছোট/বড় হলে (যেমন sorted input-এ প্রথম element pivot)।

27. কোন sorting algorithm সব ক্ষেত্রে O(n log n) এবং stable?

(ক) Selection sort
(খ) Heap sort
(গ) Merge sort
(ঘ) Quick sort

উত্তর: (গ) — heap sort O(n log n) কিন্তু stable নয়।

28. Swap flag-সহ optimized bubble sort-এর best-case complexity?

(ক) $O(n^2)$
(খ) O(1)
(গ) O(n)
(ঘ) O(n log n)

উত্তর: (গ) — আগে থেকেই sorted হলে এক pass-এ কোনো swap নেই, থেমে যায়।

29. Binary Search Tree-এর কোন traversal sorted ক্রমে উপাদান দেয়?

(ক) Preorder
(খ) Inorder
(গ) Level order
(ঘ) Postorder

উত্তর: (খ) — Left → Root → Right।

30. Singly linked list-এর শুরুতে নতুন node যোগ করার complexity?

(ক) $O(n^2)$
(খ) O(log n)
(গ) O(n)
(ঘ) O(1)

উত্তর: (ঘ) — শুধু head পয়েন্টার বদলাতে হয়।

31. BFS (Breadth First Search)-এ কোন ডেটা স্ট্রাকচার ব্যবহৃত হয়?

(ক) Queue
(খ) Heap
(গ) Stack
(ঘ) Tree

উত্তর: (ক) — DFS-এ stack (বা recursion)।

32. Infix `A + B * C`-এর postfix রূপ?

(ক) +A*BC
(খ) AB+C*
(গ) AB*C+
(ঘ) ABC*+

উত্তর: (ঘ) — * এর precedence বেশি: B C * আগে, তারপর A যোগ।

33. Postfix `2 3 4 * +`-এর মান?

(ক) 20
(খ) 14
(গ) 24
(ঘ) 9

উত্তর: (খ) — 3 × 4 = 12, তারপর 2 + 12 = 14।

34. Dijkstra's algorithm কোন ক্ষেত্রে সঠিক ফল নাও দিতে পারে?

(ক) Negative weight edge
(খ) Directed graph
(গ) Weighted graph
(ঘ) Disconnected graph

উত্তর: (ক) — negative edge-এর জন্য Bellman-Ford লাগে।

35. নিচের কোনটি hashing-এ collision resolution পদ্ধতি নয়?

(ক) Quadratic probing
(খ) Binary search
(গ) Linear probing
(ঘ) Chaining

উত্তর: (খ)

36. Hash table size 13, h(k) = k mod 13, linear probing; আগে 3 আছে। 16 কোন slot-এ বসবে?

(ক) 4
(খ) 16
(গ) 3
(ঘ) 5

উত্তর: (ক) — 16 mod 13 = 3, ভরা ⇒ পরের ফাঁকা slot 4 (BB AP 2023-এর প্রশ্নের অংশ)।

37. n-টি vertex-যুক্ত connected graph-এর spanning tree-তে edge সংখ্যা?

(ক) n − 1
(খ) n + 1
(গ) n
(ঘ) n(n − 1)/2

উত্তর: (ক)

38. Binary max-heap থেকে root (maximum) extract করার complexity?

(ক) O(n log n)
(খ) O(1)
(গ) O(n)
(ঘ) O(log n)

উত্তর: (ঘ) — শেষ element root-এ এনে heapify-down, উচ্চতা log n।

39. Dynamic programming-এর কোন পদ্ধতি ছোট subproblem থেকে শুরু করে টেবিল পূরণ করে?

(ক) Top-down
(খ) Backtracking
(গ) Bottom-up
(ঘ) Greedy

উত্তর: (গ) — Bottom-up বা tabulation (BIBM 2026 MCQ)।

40. দুটি stack দিয়ে queue বানালে enqueue-এর amortized complexity?

(ক) O(1)
(খ) O(log n)
(গ) $O(n^2)$
(ঘ) O(n)

উত্তর: (ক) — প্রতিটি element সর্বোচ্চ দুবার স্থানান্তর হয়, তাই গড়ে O(1) (AP ১৭.০১.২০২৬-এর লিখিত প্রশ্নের ধারণা)।

৬.৩ DBMS ও SQL (৪১–৫৮)

41. GROUP BY-এর পরে aggregate দিয়ে group filter করতে কোন clause?

(ক) HAVING
(খ) WHERE
(গ) WITH
(ঘ) ORDER BY

উত্তর: (ক)

42. কোন কমান্ড টেবিলের structure ও data দুটোই মুছে দেয়?

(ক) DELETE
(খ) REMOVE
(গ) DROP
(ঘ) TRUNCATE

উত্তর: (গ)

43. TRUNCATE কোন শ্রেণির কমান্ড?

(ক) DML
(খ) DCL
(গ) TCL
(ঘ) DDL

উত্তর: (ঘ)

44. GRANT ও REVOKE কোন শ্রেণির?

(ক) DML
(খ) TCL
(গ) DCL
(ঘ) DDL

উত্তর: (গ) — Data Control Language।

45. Transaction শেষ করে পরিবর্তন স্থায়ী করার কমান্ড?

(ক) COMMIT
(খ) END
(গ) ROLLBACK
(ঘ) SAVE

উত্তর: (ক) — ROLLBACK পরিবর্তন বাতিল করে; দুটোই TCL।

46. ACID-এর "A" দিয়ে কী বোঝায়?

(ক) Accuracy
(খ) Availability
(গ) Atomicity
(ঘ) Authorization

উত্তর: (গ) — all or nothing।

47. Primary key-এর ক্ষেত্রে কোনটি সত্য?

(ক) একটি টেবিলে একাধিক primary key থাকে
(খ) Duplicate হতে পারে
(গ) NULL হতে পারে
(ঘ) Unique ও NOT NULL

উত্তর: (ঘ)

48. Foreign key কী নিশ্চিত করে?

(ক) Domain integrity
(খ) Entity integrity
(গ) Referential integrity
(ঘ) Normalization

উত্তর: (গ)

49. 2NF কোন dependency দূর করে?

(ক) Multivalued
(খ) Join
(গ) Transitive
(ঘ) Partial

উত্তর: (ঘ) — 3NF দূর করে transitive dependency।

50. Many-to-many relationship কীভাবে বাস্তবায়ন করা হয়?

(ক) Index দিয়ে
(খ) View দিয়ে
(গ) এক টেবিলে FK দিয়ে
(ঘ) Junction (associative) টেবিল দিয়ে

উত্তর: (ঘ) — দুই টেবিলের PK নিয়ে composite key।

51. বাম টেবিলের সব row এবং ডান টেবিলের মিলে যাওয়া row দেয় কোন join?

(ক) RIGHT JOIN
(খ) CROSS JOIN
(গ) INNER JOIN
(ঘ) LEFT JOIN

উত্তর: (ঘ) — না মিললে ডানের কলামে NULL।

52. ফলাফল থেকে duplicate row বাদ দিতে কোন keyword?

(ক) SINGLE
(খ) DISTINCT
(গ) DIFFERENT
(ঘ) UNIQUE

উত্তর: (খ)

53. LIKE অপারেটরে যেকোনো সংখ্যক অক্ষর বোঝাতে কোন wildcard?

(ক) *
(খ) _
(গ) %
(ঘ) ?

উত্তর: (গ) — _ মানে ঠিক একটি অক্ষর।

54. ছাত্রদের নম্বর অনুযায়ী বড় থেকে ছোট সাজাতে কোনটি সঠিক?

(ক) ORDER BY marks DESC
(খ) ORDER BY marks ASC
(গ) SORT BY marks DESC
(ঘ) GROUP BY marks DESC

উত্তর: (ক)

55. `COUNT(*)` কী গণনা করে?

(ক) শুধু NULL নয় এমন মান
(খ) NULL-সহ সব row
(গ) শুধু distinct মান
(ঘ) কলাম সংখ্যা

উত্তর: (খ) — COUNT(column) NULL বাদ দেয়।

56. নিচের কোনটি aggregate function নয়?

(ক) COUNT
(খ) AVG
(গ) SUM
(ঘ) UPPER

উত্তর: (ঘ) — UPPER একটি scalar (string) function।

57. Index তৈরির ফলে সাধারণত কী ঘটে?

(ক) স্টোরেজ কমে
(খ) INSERT দ্রুত, SELECT ধীর
(গ) SELECT দ্রুত, INSERT/UPDATE কিছুটা ধীর
(ঘ) সব অপারেশন ধীর

উত্তর: (গ)

58. `CHECK (balance >= 0)` constraint ACID-এর কোন বৈশিষ্ট্য রক্ষায় সাহায্য করে?

(ক) Isolation
(খ) Durability
(গ) Atomicity
(ঘ) Consistency

উত্তর: (ঘ) — নিয়ম ভঙ্গকারী transaction-কে DB গ্রহণ করে না।

৬.৪ Operating System (৫৯–৭২)

59. কোন scheduling algorithm সর্বনিম্ন গড় waiting time দেয়?

(ক) SJF
(খ) FCFS
(গ) Priority
(ঘ) Round Robin

উত্তর: (ক)

60. Round Robin scheduling-এর মূল প্যারামিটার?

(ক) Burst estimate
(খ) Priority
(গ) Time quantum
(ঘ) Arrival order শুধু

উত্তর: (গ)

61. একই process-এর thread-গুলো কোনটি share করে না?

(ক) Stack
(খ) Heap
(গ) Code
(ঘ) Open files

উত্তর: (ক) — প্রতিটি thread-এর নিজস্ব stack ও register।

62. Deadlock হওয়ার জন্য কয়টি শর্ত একসাথে থাকা প্রয়োজন?

(ক) 3
(খ) 5
(গ) 4
(ঘ) 2

উত্তর: (গ) — mutual exclusion, hold & wait, no preemption, circular wait।

63. Banker's algorithm কীসের জন্য?

(ক) Deadlock avoidance
(খ) Deadlock detection
(গ) CPU scheduling
(ঘ) Deadlock prevention

উত্তর: (ক) — safe state যাচাই করে resource দেয়।

64. Thrashing বলতে কী বোঝায়?

(ক) Execution-এর চেয়ে paging-এ বেশি সময় ব্যয়
(খ) Deadlock
(গ) Disk পূর্ণ হয়ে যাওয়া
(ঘ) CPU অতিরিক্ত গরম

উত্তর: (ক)

65. Belady's anomaly কোন page replacement algorithm-এ দেখা যায়?

(ক) LFU
(খ) LRU
(গ) Optimal
(ঘ) FIFO

উত্তর: (ঘ) — frame বাড়ালেও page fault বাড়তে পারে।

66. Producer–consumer সমস্যা মূলত কোন ধরনের সমস্যা?

(ক) Disk scheduling
(খ) Paging
(গ) Process synchronization
(ঘ) Memory allocation

উত্তর: (গ) — bounded buffer; mutex + empty + full semaphore দিয়ে সমাধান।

67. দুটি process যেন একসাথে critical section-এ না ঢোকে — এই শর্তকে কী বলে?

(ক) Starvation
(খ) Race condition
(গ) Synchronous condition
(ঘ) Mutual exclusion

উত্তর: (ঘ) — BIBM 2026 MCQ।

68. Context switching কোথায় ঘটে?

(ক) Compiler-এ
(খ) Hard disk-এ
(গ) CPU-তে
(ঘ) I/O device-এ

উত্তর: (গ) — CPU এক process থেকে আরেক process-এ যায়, state PCB-তে রাখা হয়।

69. Computer boot প্রক্রিয়ার সঠিক ক্রম কোনটি?

(ক) MBR → POST → Login → Kernel
(খ) Drivers → POST → Kernel → Login
(গ) Kernel → POST → Bootloader → Login
(ঘ) POST → MBR/GPT → Kernel → Drivers → Login

উত্তর: (ঘ) — AP ০৩.০১.২০২৬-এর লিখিত প্রশ্ন।

70. Linux-এ ফাইলের permission পরিবর্তনের কমান্ড?

(ক) chown
(খ) chgrp
(গ) chmod
(ঘ) umask

উত্তর: (গ) — chown owner বদলায়।

71. `chmod 644 file` দিলে group-এর permission কী?

(ক) rw-
(খ) r--
(গ) r-x
(ঘ) rwx

উত্তর: (খ) — 6 = rw- (owner), 4 = r-- (group), 4 = r-- (others)।

72. CPU-র জন্য অপেক্ষারত কিন্তু memory-তে থাকা process কোন state-এ?

(ক) New
(খ) Terminated
(গ) Ready
(ঘ) Waiting

উত্তর: (গ) — I/O-র জন্য অপেক্ষা করলে Waiting/Blocked।

৬.৫ Networking (৭৩–৯০)

73. OSI মডেলে কয়টি লেয়ার?

(ক) 5 (খ) 8
(গ) 7 (ঘ) 4

উত্তর: (গ)

74. Routing কোন OSI লেয়ারের কাজ?

(ক) Data Link (খ) Session
(গ) Transport (ঘ) Network

উত্তর: (ঘ)

75. Transport লেয়ারে TCP-র PDU-এর নাম?

(ক) Packet (খ) Bit
(গ) Segment (ঘ) Frame

উত্তর: (গ) — Network: packet, Data link: frame।

76. HTTPS-এর default port?

(ক) 8080 (খ) 443
(গ) 22 (ঘ) 80

উত্তর: (খ)

77. SSH-এর standard TCP port?

(ক) 21 (খ) 25
(গ) 23 (ঘ) 22

উত্তর: (ঘ) — 21 FTP, 23 Telnet, 25 SMTP।

78. DNS কোন port ব্যবহার করে?

(ক) 110 (খ) 143
(গ) 53 (ঘ) 67

উত্তর: (গ)

79. ইমেইল গ্রহণ (receive) করতে কোন প্রোটোকল?

(ক) SNMP (খ) POP3
(গ) FTP (ঘ) SMTP

উত্তর: (খ) — IMAP-ও receive; SMTP পাঠায়।

80. IPv4 ও IPv6 address যথাক্রমে কত বিটের?

(ক) 64 ও 128 (খ) 32 ও 64
(গ) 16 ও 32 (ঘ) 32 ও 128

উত্তর: (ঘ)

81. কোনটি private IP address?

(ক) 172.32.1.1 (খ) 192.168.10.5
(গ) 11.0.0.1 (ঘ) 8.8.8.8

উত্তর: (খ) — private: 10.0.0.0/8, 172.16.0.0–172.31.255.255, 192.168.0.0/16।

82. /26 সাবনেটে usable host কয়টি?

(ক) 126 (খ) 62
(গ) 30 (ঘ) 64

উত্তর: (খ) — host bit 6: $2^6 - 2 = 62$।

83. /27-এর subnet mask?

(ক) 255.255.255.192 (খ) 255.255.255.240
(গ) 255.255.255.224 (ঘ) 255.255.255.128

উত্তর: (গ) — শেষ অক্টেটে 3টি 1: 128 + 64 + 32 = 224।

84. DHCP-র চার ধাপের সঠিক ক্রম?

(ক) Discover–Request–Offer–Ack (খ) Offer–Discover–Ack–Request
(গ) Discover–Offer–Request–Ack (ঘ) Request–Offer–Discover–Ack

উত্তর: (গ) — DORA।

85. ARP কী কাজ করে?

(ক) IP → MAC (খ) IP বরাদ্দ করে
(গ) MAC → Port (ঘ) Domain → IP

উত্তর: (ক)

86. TCP 3-way handshake-এর সঠিক ক্রম?

(ক) SYN, SYN-ACK, ACK (খ) SYN-ACK, SYN, ACK
(গ) ACK, SYN, FIN (ঘ) SYN, ACK, SYN-ACK

উত্তর: (ক)

87. Real-time streaming-এ বেশি ব্যবহৃত connectionless প্রোটোকল?

(ক) UDP (খ) TCP
(গ) FTP (ঘ) SMTP

উত্তর: (ক) — কম overhead, retransmission নেই।

88. OSPF কোন algorithm ব্যবহার করে?

(ক) Kruskal (খ) Prim
(গ) Dijkstra (SPF) (ঘ) Bellman-Ford

উত্তর: (গ) — RIP ব্যবহার করে Bellman-Ford, max hop 15।

89. MAC address দেখে frame forward করে কোন ডিভাইস (Layer 2)?

(ক) Router (খ) Switch
(গ) Hub (ঘ) Repeater

উত্তর: (খ)

90. 8.8.8.8-এ ping হয় কিন্তু www.google.com খোলে না — সম্ভাব্য কারণ?

(ক) DHCP ব্যর্থ
(খ) ক্যাবল বিচ্ছিন্ন
(গ) DNS সমস্যা
(ঘ) Gateway নেই

উত্তর: (গ) — IP connectivity আছে, name resolution হচ্ছে না (BIBM 2026 MCQ)।

৬.৬ Security (৯১–১০২)

91. CIA triad-এ "I" মানে?

(ক) Integrity
(খ) Isolation
(গ) Information
(ঘ) Identity

উত্তর: (ক)

92. প্রেরক পরে বার্তা পাঠানোর কথা অস্বীকার করতে না পারার নীতি?

(ক) Availability
(খ) Confidentiality
(গ) Non-repudiation
(ঘ) Integrity

উত্তর: (গ) — digital signature দিয়ে নিশ্চিত হয়।

93. কোনটি asymmetric encryption algorithm?

(ক) RSA
(খ) DES
(গ) AES
(ঘ) 3DES

উত্তর: (ক)

94. কোনটি cryptographic hash function?

(ক) RSA-2048
(খ) AES-256
(গ) SHA-256
(ঘ) Base64

উত্তর: (গ) — Base64 শুধু encoding।

95. "ব্যাংক থেকে বলছি, আপনার OTP বলুন" — এটি কোন আক্রমণ?

(ক) Social engineering (vishing)
(খ) DDoS
(গ) SQL injection
(ঘ) Brute force

উত্তর: (ক)

96. SQL injection প্রতিরোধের সবচেয়ে কার্যকর উপায়?

(ক) Antivirus
(খ) HTTPS ব্যবহার
(গ) Prepared statement / parameterized query
(ঘ) বড় পাসওয়ার্ড

উত্তর: (গ)

97. DoS/DDoS আক্রমণ মূলত CIA-এর কোনটিকে আঘাত করে?

(ক) Confidentiality
(খ) Availability
(গ) Integrity
(ঘ) Authentication

উত্তর: (খ)

98. সন্দেহজনক ট্রাফিক শনাক্ত করে inline-এ স্বয়ংক্রিয়ভাবে ব্লক করে কোনটি?

(ক) Hub
(খ) Proxy
(গ) IPS
(ঘ) IDS

উত্তর: (গ) — IDS শুধু alert দেয়।

99. DMZ-এর মূল উদ্দেশ্য?

(ক) Backup সংরক্ষণ
(খ) IP বরাদ্দ
(গ) LAN-এর গতি বাড়ানো
(ঘ) Public-facing server-কে আলাদা নিরাপদ জোনে রাখা

উত্তর: (ঘ)

100. Password + OTP + fingerprint ব্যবহারকে কী বলে?

(ক) Encryption
(খ) Multi-factor authentication
(গ) Authorization
(ঘ) Single sign-on

উত্তর: (খ) — know + have + are।

101. ব্যবহারকারীকে কাজের জন্য ন্যূনতম প্রয়োজনীয় access দেওয়ার নীতি?

(ক) Least privilege
(খ) Separation of duties
(গ) Zero day
(ঘ) Defense in depth

উত্তর: (ক)

102. Security awareness training কোন ধরনের control?

(ক) Detective only
(খ) Technical
(গ) Administrative
(ঘ) Physical

উত্তর: (গ) — BIBM 2026 MCQ।

৬.৭ SE, Web, Cloud ও AI (১০৩–১১০)

103. Linear sequential SDLC model কোনটি?

(ক) Waterfall
(খ) Agile
(গ) Spiral
(ঘ) Prototype

উত্তর: (ক)

104. Git-এ local commit রিমোট রিপোজিটরিতে পাঠানোর কমান্ড?

(ক) git commit
(খ) git pull
(গ) git push
(ঘ) git clone

উত্তর: (গ)

105. HTTP status 404-এর অর্থ?

(ক) Internal Server Error
(খ) Forbidden
(গ) Bad Request
(ঘ) Not Found

উত্তর: (ঘ)

106. Heroku বা Google App Engine কোন cloud service model?

(ক) IaaS
(খ) DaaS
(গ) PaaS
(ঘ) SaaS

উত্তর: (গ)

107. সরাসরি hardware-এর উপর চলে (bare-metal) কোন hypervisor?

(ক) Type 2
(খ) Container
(গ) Type 1
(ঘ) Emulator

উত্তর: (গ) — যেমন VMware ESXi, Hyper-V।

108. Label ছাড়া ডেটা থেকে গ্রাহকদের দলে ভাগ (clustering) কোন ধরনের learning?

(ক) Reinforcement
(খ) Supervised
(গ) Transfer
(ঘ) Unsupervised

উত্তর: (ঘ)

109. A* search-এর evaluation function?

(ক) f(n) = g(n)
(খ) f(n) = h(n)
(গ) f(n) = g(n) × h(n)
(ঘ) f(n) = g(n) + h(n)

উত্তর: (ঘ) — শুধু h(n) হলে greedy best-first।

110. কোড না দেখে শুধু requirement অনুযায়ী টেস্ট করাকে কী বলে?

(ক) Code review
(খ) Black-box testing
(গ) Unit testing
(ঘ) White-box testing

উত্তর: (খ)

৬.৮ গণিত (১১১–১৩৫)

111. 250-এর 20% + 400-এর 15% = ?

(ক) 105
(খ) 110
(গ) 120
(ঘ) 100

উত্তর: (খ) — 50 + 60 = 110।

112. একটি সংখ্যা 20% বাড়িয়ে পরে 20% কমালে মোট পরিবর্তন?

(ক) পরিবর্তন নেই
(খ) 4% কম
(গ) 4% বেশি
(ঘ) 2% কম

উত্তর: (খ) — 1.2 × 0.8 = 0.96।

113. ক্রয়মূল্য 800 টাকা, বিক্রয়মূল্য 920 টাকা — লাভ কত শতাংশ?

(ক) 20%
(খ) 18%
(গ) 15%
(ঘ) 12%

উত্তর: (গ) — 120/800 × 100 = 15%।

114. 1000 টাকার পণ্যে পরপর 10% ও 5% ছাড়ে বিক্রয়মূল্য?

(ক) 850 (খ) 860

(গ) 845 (ঘ) 855

উত্তর: (ঘ) — 1000 × 0.9 × 0.95 = 855।

115. 20% ছাড় দিয়েও 20% লাভ হলে marked price ক্রয়মূল্যের চেয়ে কত % বেশি?

(ক) 20% (খ) 40%

(গ) 30% (ঘ) 50%

উত্তর: (ঘ) — CP 100, SP 120 = 0.8 MP ⇒ MP = 150 (BIBM 2026 MCQ)।

116. A : B = 2 : 3 এবং B : C = 4 : 5 হলে A : B : C = ?

(ক) 2 : 3 : 5 (খ) 8 : 12 : 15

(গ) 4 : 6 : 5 (ঘ) 8 : 12 : 10

উত্তর: (খ) — B-কে 12 করতে: A = 8, C = 15।

117. A একটি কাজ 12 দিনে, B 18 দিনে করে। একসাথে কত দিনে?

(ক) 7.5 দিন (খ) 15 দিন

(গ) 7.2 দিন (ঘ) 6 দিন

উত্তর: (গ) — 1/12 + 1/18 = 5/36 ⇒ 36/5 = 7.2।

118. 36 km/h = কত m/s?

(ক) 6 (খ) 18

(গ) 12 (ঘ) 10

উত্তর: (ঘ) — 36 × 5/18 = 10।

119. স্থির পানিতে নৌকার বেগ 10 km/h, স্রোত 2 km/h। ভাটিতে 60 km যেতে সময়?

(ক) 4 ঘণ্টা (খ) 7.5 ঘণ্টা

(গ) 5 ঘণ্টা (ঘ) 6 ঘণ্টা

উত্তর: (গ) — ভাটির বেগ 12 km/h; 60/12 = 5।

120. 150 m লম্বা ট্রেন একটি খুঁটি 10 সেকেন্ডে অতিক্রম করে। ট্রেনের বেগ?

(ক) 60 km/h (খ) 15 km/h

(গ) 54 km/h (ঘ) 45 km/h

উত্তর: (গ) — 15 m/s × 18/5 = 54 km/h।

121. $(a + b)^2 = 49$ এবং $ab = 6$ হলে $a^2 + b^2 = ?$

(ক) 61 (খ) 37

(গ) 43 (ঘ) 55

উত্তর: (খ) — 49 − 2 × 6 = 37।

122. $x - 1/x = 3$ হলে $x^2 + 1/x^2 = ?$

(ক) 7 (খ) 9

(গ) 13 (ঘ) 11

উত্তর: (ঘ) — $(x - 1/x)^2 + 2 = 9 + 2 = 11$।

123. a + b = 5 এবং a − b = 1 হলে $a^2 - b^2$ = ?

(ক) 4 (খ) 24

(গ) 6 (ঘ) 5

উত্তর: (ঘ) — (a + b)(a − b) = 5 × 1।

124. 7 cm ব্যাসার্ধের বৃত্তের ক্ষেত্রফল (π = 22/7)?

(ক) 144 cm^2 (খ) 154 cm^2

(গ) 44 cm^2 (ঘ) 308 cm^2

উত্তর: (খ) — 22/7 × 49 = 154।

125. 3 cm বাহুর ঘনকের সমগ্র পৃষ্ঠতলের ক্ষেত্রফল?

(ক) 54 cm^2 (খ) 27 cm^2

(গ) 18 cm^2 (ঘ) 36 cm^2

উত্তর: (ক) — $6a^2 = 6 \times 9 = 54$।

126. 6, 8, 10 cm বাহুবিশিষ্ট ত্রিভুজের ক্ষেত্রফল?

(ক) 48 cm^2 (খ) 40 cm^2

(গ) 30 cm^2 (ঘ) 24 cm^2

উত্তর: (ঘ) — $6^2 + 8^2 = 10^2$, সমকোণী; ½ × 6 × 8 = 24।

127. দুটি ছক্কা একসাথে নিক্ষেপে যোগফল 8 হওয়ার সম্ভাবনা?

(ক) 1/9 (খ) 5/36

(গ) 1/6 (ঘ) 7/36

উত্তর: (খ) — (2,6),(3,5),(4,4),(5,3),(6,2) = 5টি।

128. ৫২টি তাসের প্যাকেট থেকে একটি টেক্কা (Ace) পাওয়ার সম্ভাবনা?

(ক) 1/4 (খ) 4/13

(গ) 1/52 (ঘ) 1/13

উত্তর: (ঘ) — 4/52।

129. একটি মুদ্রা ৩ বার নিক্ষেপে অন্তত একটি Head-এর সম্ভাবনা?

(ক) 1/8 (খ) 3/8

(গ) 1/2 (ঘ) 7/8

উত্তর: (ঘ) — $1 - (1/2)^3$।

130. ধারাটির পরের পদ: 2, 6, 12, 20, 30, ?

(ক) 44 (খ) 36

(গ) 40 (ঘ) 42

উত্তর: (ঘ) — n(n + 1): 6 × 7 = 42।

131. ধারাটির পরের পদ: 1, 2, 4, 7, 11, ?

(ক) 15 (খ) 16

(গ) 14 (ঘ) 17

উত্তর: (খ) — পার্থক্য 1, 2, 3, 4, 5 ⇒ 11 + 5 = 16 (AP 2024-এর series প্রোগ্রাম)।

132. পিতার বয়স পুত্রের ৩ গুণ; ১০ বছর পর হবে ২ গুণ। পুত্রের বর্তমান বয়স?

(ক) 10 (খ) 15

(গ) 20 (ঘ) 12

উত্তর: (ক) — $3S + 10 = 2(S + 10) \Rightarrow S = 10$।

133. 5000 টাকার 8% হারে 3 বছরের সরল মুনাফা?

(ক) 1300 (খ) 1250

(গ) 1200 (ঘ) 1000

উত্তর: (গ) — $5000 \times 0.08 \times 3$।

134. 10000 টাকার 10% হারে 2 বছরের চক্রবৃদ্ধি মুনাফা?

(ক) 2000 (খ) 1900

(গ) 2200 (ঘ) 2100

উত্তর: (ঘ) — $10000 \times (1.21 - 1)$।

135. 12, 18 ও 30-এর লসাগু?

(ক) 120 (খ) 90

(গ) 360 (ঘ) 180

উত্তর: (ঘ) — $2^2 \times 3^2 \times 5 = 180$।

৬.৯ সাধারণ জ্ঞান — ব্যাংকিং ও আইসিটি (১৩৬–১৫০)

136. NPSB-এর পূর্ণরূপ?

(ক) National Postal Savings Bank (খ) National Payment Switch Bangladesh

(গ) National Payment System Bangladesh (ঘ) Network Payment Service Bank

উত্তর: (খ)

137. RTGS-এর পূর্ণরূপ?

(ক) Rapid Transfer Gross System (খ) Real Time Gross Settlement

(গ) Real Transaction Global Settlement (ঘ) Real Time General Service

উত্তর: (খ) — প্রতিটি লেনদেন আলাদাভাবে রিয়েল টাইমে নিষ্পত্তি।

138. BEFTN-এর পূর্ণরূপ?

(ক) Bangladesh E-Finance Transfer Network (খ) Bangladesh Electronic Financial Trade Network

(গ) Bank Electronic Fund Transaction Node (ঘ) Bangladesh Electronic Funds Transfer Network

উত্তর: (ঘ)

139. বাংলাদেশ ব্যাংকের প্রধান কার্যালয় কোথায়?

(ক) মতিঝিল, ঢাকা (খ) আগারগাঁও, ঢাকা

(গ) চট্টগ্রাম (ঘ) গুলশান, ঢাকা

উত্তর: (ক) — মতিঝিল বাণিজ্যিক এলাকা, ঢাকা-১০০০ (বিজ্ঞপ্তির ঠিকানাতেও "মতিঝিল, ঢাকা")।

140. দেশের প্রথম বেসরকারি বাণিজ্যিক ব্যাংক?

(ক) পূবালী ব্যাংক
(খ) এবি ব্যাংক
(গ) ইসলামী ব্যাংক
(ঘ) ন্যাশনাল ব্যাংক

উত্তর: (খ) — ১৯৮২ (তৎকালীন Arab Bangladesh Bank)।

141. বাংলাদেশে প্রথম কম্পিউটার কোথায় স্থাপিত হয়?

(ক) BCSIR
(খ) বুয়েট
(গ) পরমাণু শক্তি কেন্দ্র, ঢাকা
(ঘ) ঢাকা বিশ্ববিদ্যালয়

উত্তর: (গ) — IBM 1620, ১৯৬৪।

142. বাংলাদেশ প্রথম কোন সাবমেরিন কেবলের সাথে যুক্ত হয়?

(ক) SEA-ME-WE 3
(খ) FLAG
(গ) SEA-ME-WE 5
(ঘ) SEA-ME-WE 4

উত্তর: (ঘ) — ২০০৬, কক্সবাজার ল্যান্ডিং স্টেশন।

143. বঙ্গবন্ধু স্যাটেলাইট-১ কোন সালে উৎক্ষেপণ করা হয়?

(ক) 2018
(খ) 2019
(গ) 2016
(ঘ) 2017

উত্তর: (ক) — ১২ মে ২০১৮।

144. ২০১৬ সালের বাংলাদেশ ব্যাংক রিজার্ভ চুরিতে কোন বার্তা নেটওয়ার্ক অপব্যবহার করা হয়?

(ক) NPSB
(খ) SWIFT
(গ) BEFTN
(ঘ) VISA

উত্তর: (খ) — নিউইয়র্ক ফেডের অ্যাকাউন্ট থেকে ভুয়া SWIFT বার্তায়।

145. ওই রিজার্ভ চুরির ৮১ মিলিয়ন ডলার মূলত কোন দেশের ব্যাংকে যায়?

(ক) হংকং
(খ) মালয়েশিয়া
(গ) শ্রীলঙ্কা
(ঘ) ফিলিপাইন

উত্তর: (ঘ) — RCBC ব্যাংকের মাধ্যমে।

146. SWIFT-এর সদর দপ্তর কোন দেশে?

(ক) সুইজারল্যান্ড
(খ) যুক্তরাজ্য
(গ) বেলজিয়াম
(ঘ) যুক্তরাষ্ট্র

উত্তর: (গ) — La Hulpe, Belgium।

147. Bank for International Settlements (BIS)-এর সদর দপ্তর?

(ক) ব্রাসেলস
(খ) প্যারিস
(গ) বাসেল
(ঘ) জেনেভা

উত্তর: (গ) — বাসেল, সুইজারল্যান্ড (Basel Accords এ কারণেই)।

148. IMF-এর সদর দপ্তর?

(ক) ওয়াশিংটন ডি.সি.
(খ) জেনেভা
(গ) লন্ডন
(ঘ) নিউইয়র্ক

উত্তর: (ক)

149. CIRDAP-এর সদর দপ্তর কোথায়?

(ক) ঢাকা (খ) ম্যানিলা

(গ) কাঠমান্ডু (ঘ) ব্যাংকক

উত্তর: (ক) — AP ০৩.০১.২০২৬-এর GK-তে এসেছিল।

150. WIPO-এর পূর্ণরূপ?

(ক) World International Patent Office (খ) World Information Policy Organization

(গ) World Investment Promotion Organization (ঘ) World Intellectual Property Organization

উত্তর: (ঘ) — সদর দপ্তর জেনেভা; AP ০৩.০১.২০২৬-এর GK।

৬.১০ উত্তরপত্র (দ্রুত মিলানোর জন্য)

১–১৫	১৬–৩০	৩১–৪৫	৪৬–৬০	৬১–৭৫
1 ঘ · 2 খ · 3 খ · 4 ক · 5 খ · 6 ঘ · 7 গ · 8 ঘ · 9 ঘ · 10 ক · 11 খ · 12 গ · 13 ঘ · 14 ঘ · 15 ক	16 ঘ · 17 খ · 18 খ · 19 ক · 20 খ · 21 ঘ · 22 গ · 23 খ · 24 গ · 25 ক · 26 ঘ · 27 গ · 28 গ · 29 খ · 30 ঘ	31 ক · 32 ঘ · 33 খ · 34 ক · 35 খ · 36 ক · 37 ক · 38 ঘ · 39 গ · 40 ক · 41 ক · 42 গ · 43 ঘ · 44 গ · 45 ক	46 গ · 47 ঘ · 48 গ · 49 ঘ · 50 ঘ · 51 ঘ · 52 খ · 53 গ · 54 ক · 55 খ · 56 ঘ · 57 গ · 58 ঘ · 59 ক · 60 গ	61 ক · 62 গ · 63 ক · 64 ক · 65 ঘ · 66 গ · 67 ঘ · 68 গ · 69 ঘ · 70 গ · 71 খ · 72 গ · 73 গ · 74 ঘ · 75 গ
৭৬–৯০	**৯১–১০৫**	**১০৬–১২০**	**১২১–১৩৫**	**১৩৬–১৫০**
76 খ · 77 ঘ · 78 গ · 79 খ · 80 ঘ · 81 খ · 82 খ · 83 গ · 84 গ · 85 ক · 86 ক · 87 ক · 88 গ · 89 খ · 90 গ	91 ক · 92 গ · 93 ক · 94 গ · 95 ক · 96 গ · 97 খ · 98 গ · 99 ঘ · 100 খ · 101 ক · 102 গ · 103 ক · 104 গ · 105 ঘ	106 গ · 107 গ · 108 ঘ · 109 ঘ · 110 খ · 111 খ · 112 খ · 113 গ · 114 ঘ · 115 ঘ · 116 খ · 117 গ · 118 ঘ · 119 গ · 120 গ	121 খ · 122 ঘ · 123 ঘ · 124 খ · 125 ক · 126 ঘ · 127 খ · 128 ঘ · 129 ঘ · 130 ঘ · 131 খ · 132 ক · 133 গ · 134 ঘ · 135 ঘ	136 খ · 137 খ · 138 ঘ · 139 ক · 140 খ · 141 গ · 142 ঘ · 143 ক · 144 খ · 145 ঘ · 146 গ · 147 গ · 148 ক · 149 ক · 150 ঘ

অধ্যায় ৭ — গণিত, English Language Proficiency ও GK

Mathematics (MCQ 15 + Written 20) · English (Written 30) · General Knowledge (MCQ 10)

এই অংশে মোট **৭৫ নম্বর** (MCQ ২৫ + লিখিত ৫০)। টেকনিক্যাল অংশে সবাই কাছাকাছি নম্বর পায়, তাই এখানের নম্বরই অনেক সময় মেধাতালিকা ঠিক করে দেয়। বিগত AP পেপারের সব প্রশ্নের সমাধান আগে, তারপর অনুশীলন।

৭.১ গণিত — বিগত AP প্রশ্নের পূর্ণ সমাধান

প্রশ্ন 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.

বর্গের বাহু = 32/4 = 8 m ⇒ ক্ষেত্রফল = 64 m^2

আয়তের দৈর্ঘ্য = 64/4 = **16 m**

প্রশ্ন E · AP 17.01.2026

If $2 < x < 5$ and $3 < y < 5$, find the range of xy.

x ও y দুটোই ধনাত্মক, তাই সীমাগুলো গুণ করা যায়: সর্বনিম্ন 2 × 3 = 6, সর্বোচ্চ 5 × 5 = 25 ⇒ **$6 < xy < 25$**

সতর্কতা: কোনো চলক ঋণাত্মক হতে পারলে এভাবে সরাসরি গুণ করা যায় না — তখন চার কোণের গুণফল দেখে সর্বনিম্ন/সর্বোচ্চ বের করতে হয়।

প্রশ্ন 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.

মেয়ে = g, ছেলে = 1.35g ⇒ 2.35g = 94 ⇒ g = 40, ছেলে = 54

নতুন ছেলে = 54 + 6 = 60 ⇒ অনুপাত = 60 : 40 = **3 : 2**

প্রশ্ন 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.

একই দূরত্বে সময় তিনগুণ ⇒ গতি এক-তৃতীয়াংশ ⇒ 12 + s = 3(12 − s)

12 + s = 36 − 3s ⇒ 4s = 24 ⇒ **স্রোতের বেগ 6 km/h** (যাচাই: ভাটিতে 18, উজানে 6 — অনুপাত 3:1 ✔)

প্রশ্ন 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**

প্রশ্ন 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%?

(সংকলনে শতাংশ অস্পষ্ট; প্রচলিত প্রশ্নে 16⅔%।)

24 = মোটের 1/6 ⇒ মোট 144 খেলা; ৭৫% = 108 জয় ⇒ আরও **93টি** জিততে হবে

প্রশ্ন 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.

$(x + 4)(y - 2) = xy \Rightarrow x = 2y - 4$; $(x - 4)(y + 4) = xy \Rightarrow x = y + 4$

$\Rightarrow y = 8, x = 12 \Rightarrow$ মোট **96 জন**

B-এর চতুর্থ প্রশ্নটি (বর্গ PSRT-এর ক্ষেত্রফল) চিত্রনির্ভর; সংকলনে চিত্রের মাপ পাওয়া যায়নি, তাই বাদ।

লিখিত গণিতের ৪টি প্রশ্নে পূর্ণ নম্বর পেতে

উত্তরের কাঠামো

"ধরি..." দিয়ে চলক সংজ্ঞা → সমীকরণ গঠন → সমাধান → উত্তর বক্স করুন → এক লাইনে যাচাই। ক্যালকুলেটর নেই, তাই ভগ্নাংশে কাজ করুন (0.35 নয়, 7/20)।

অনুশীলনী (BIBM ধাঁচের) — সমাধানসহ

#	প্রশ্ন	সমাধান
1	A, B-কে ২০% লাভে; B, C-কে ২৫% লাভে বিক্রি করল। C দিল ১৫০০ টাকা। A-এর ক্রয়মূল্য?	$1500/(1.2 \times 1.25) =$ **1000 টাকা**
2	দুজনের বয়সের অনুপাত 3:5; ৬ বছর পর 2:3। বর্তমান বয়স?	$(3x+6)/(5x+6) = 2/3 \Rightarrow x = 6 \Rightarrow$ **18 ও 30**
3	A ১২ দিনে, B ১৮ দিনে কাজ করে। একসাথে ৪ দিন করার পর A চলে গেল। B বাকিটা কত দিনে করবে?	৪ দিনে $4(1/12 + 1/18) = 5/9$; বাকি $4/9 \times 18 =$ **8 দিন**
4	স্থির পানিতে নৌকা 15 km/h, স্রোত 3 km/h। 36 km গিয়ে ফিরে আসতে সময়?	$36/18 + 36/12 = 2 + 3 =$ **5 ঘণ্টা**
5	দাম ২৫% বাড়লে খরচ অপরিবর্তিত রাখতে ভোগ কত % কমাতে হবে?	$25/125 \times 100 =$ **20%**
6	১০,০০০ টাকায় ১০% হারে ২ বছরে চক্রবৃদ্ধি ও সরল মুনাফার পার্থক্য?	$P(r)^2 = 10000 \times 0.01 =$ **100 টাকা**
7	পরপর ৫টি বিজোড় সংখ্যার গড় 27। বৃহত্তমটি?	মাঝেরটি 27 ⇒ 23, 25, 27, 29, **31**
8	$x - y = 4$, $xy = 21$ হলে $x^2 + y^2$ ও $x^3 - y^3$?	$16 + 42 =$ **58**; $(x-y)^3 + 3xy(x-y) = 64 + 252 =$ **316**
9	বৃত্তের ক্ষেত্রফল 154 cm^2 ($\pi = 22/7$)। পরিধি?	$r^2 = 49 \Rightarrow r = 7 \Rightarrow 2\pi r =$ **44 cm**
10	দুটি ছক্কায় যোগফল ১০ বা বেশি হওয়ার সম্ভাবনা?	(4,6),(6,4),(5,5),(5,6),(6,5),(6,6) = 6/36 = **1/6**
11	একটি সভায় ৪৫টি হ্যান্ডশেক হলো। সদস্য কতজন?	$n(n-1)/2 = 45 \Rightarrow$ **n = 10**
12	2, 6, 12, 20, 30, ?	$n(n+1) \Rightarrow 6 \times 7 =$ **42**
13	150 m ট্রেন একটি খুঁটি ১০ সেকেন্ডে পার হয়। গতি km/h-এ?	$15 \text{ m/s} \times 18/5 =$ **54 km/h**
14	আয়তের দৈর্ঘ্য ২০% বাড়ল, প্রস্থ ২০% কমল। ক্ষেত্রফল?	$1.2 \times 0.8 = 0.96 \Rightarrow$ **৪% কমবে**
15	$\log_x 81 = 4$ হলে x?	$x^4 = 81 \Rightarrow$ **x = 3**

৭.২ English Language Proficiency (৩০ নম্বর)

বিগত পেপারে এই অংশ এসেছে তিনভাবে: (১) Focus + অনুবাদ (A, B), (২) Focus ২০ + Letter ১০ (C), (৩) Grammar ১০টি + Paragraph + Sentence making (E — সবচেয়ে সাম্প্রতিক AP)। তিন ধরনেরই প্রস্তুতি নিচে।

৭.২.১ Grammar — AP 17.01.2026-এর পূর্ণ সমাধান

বাক্য	উত্তর	নিয়ম
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**	"as well as"-এর আগের subject (committee, singular) অনুযায়ী verb
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**	বিপরীত ভাব ⇒ contrast connector
G. The research team arrived at (a / an) unanimous conclusion.	**a**	unanimous উচ্চারণ "ইউ" (consonant sound) ⇒ a
H. Any breach of conduct could result (in / to) suspension.	**in**	result in = পরিণতি হওয়া; result from = কারণ থেকে উদ্ভূত
I. She is one of the scholars who (advocates / advocate) a critical pedagogy approach.	**advocate**	"one of the + plural noun + who" ⇒ who-এর antecedent plural (scholars)
J. The researcher insisted that the data (be / is) analyzed before publication.	**be**	insist/suggest/demand + that ⇒ subjunctive (base form)

৭.২.২ একই ধাঁচের আরও ১০টি (উত্তরসহ)

বাক্য	উত্তর ও কারণ
Neither the manager nor the clerks (was / were) present.	**were** — neither...nor-এ কাছের subject (clerks) অনুযায়ী
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** — অবাস্তব শর্তে were
The news (is / are) encouraging.	**is** — news 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

৭.২.৩ Sentence making E · AP 17.01.2026

শব্দ (পদ)	বাক্য
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.

নিয়ম মনে রাখুন

"Do not change the grammatical category" — adjective-কে adverb (judiciously) বা verb-কে noun (fabrication) বানালে নম্বর নেই।

৭.২.৪ Paragraph — "Street Food Culture in Dhaka" E · AP 17.01.2026

মডেল (~১৭০ শব্দ)

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.

৭.২.৫ Letter (Paper C-তে ১০ নম্বর)

Formal letter-এর কাঠামো

Date → Recipient (designation, organisation, address) → Subject (এক লাইনে) → Salutation (Dear Sir/Madam,) → Body (৩ অনুচ্ছেদ: উদ্দেশ্য · বিস্তারিত · অনুরোধ/পদক্ষেপ) → Closing (Yours faithfully,) → নাম, পদবি, যোগাযোগ।

নমুনা: অনলাইন ব্যাংকিং-এ অননুমোদিত লেনদেনের অভিযোগ

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

৭.২.৬ Focus / Essay — কাঠামো ও মডেল

মডেল: 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.

বাংলা রচনা: "দুর্নীতিমুক্ত দেশ গঠনে যুবসমাজের ভূমিকা" D **— আউটলাইন**

ভূমিকা (দুর্নীতি কী, উন্নয়নের অন্তরায়) → দুর্নীতির রূপ ও ক্ষতি → যুবসমাজ কেন গুরুত্বপূর্ণ (জনসংখ্যার বড় অংশ, প্রযুক্তিদক্ষ) → ভূমিকা: নিজে সৎ থাকা, সচেতনতা ও স্বেচ্ছাসেবা, তথ্য অধিকার আইন ও ডিজিটাল সেবার ব্যবহার, সামাজিক মাধ্যমে জবাবদিহি দাবি, নেতৃত্বে সততা → প্রতিবন্ধকতা → সুপারিশ → উপসংহার।

৭.২.৭ অনুবাদ — বিগত AP প্রশ্ন

বাংলা → 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 → বাংলা 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.

অনুবাদ: 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 → বাংলা (আংশিক) 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...

অনুবাদ: বৈশ্বিক অর্থনৈতিক অস্থিরতা সারা বিশ্বের ব্যাংকিং ও আর্থিক খাতে তীব্রভাবে অনুভূত হচ্ছে। নতুন প্রযুক্তি গ্রহণের মাধ্যমে পুনরুদ্ধারের নতুন উদ্যোগ নেওয়ার ক্ষেত্রে দৃশ্যমান পরিবর্তন ঘটছে...

ব্যাংকিং ও প্রযুক্তি শব্দভান্ডার

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

৭.৩ সাধারণ জ্ঞান

বিগত AP প্রশ্ন D · BSCS 03.01.2026

প্রশ্ন	উত্তর
বাংলাদেশে প্রথম ভূমিকম্প পর্যবেক্ষণ কেন্দ্র কোথায়?	চট্টগ্রাম (১৯৫৪)
WIPO-এর পূর্ণরূপ	World Intellectual Property Organization
বাংলাদেশে VAT চালু হয় কবে?	১৯৯১ (১ জুলাই)
বাঙালি উদ্যোক্তাদের মালিকানায় প্রথম ব্যাংক	ইস্টার্ন মার্কেন্টাইল ব্যাংক (১৯৫৯; বর্তমান পূবালী ব্যাংক) — প্রচলিত উত্তর; কিছু উৎস ব্রিটিশ আমলের কুমিল্লা ব্যাংকিং কর্পোরেশনের কথাও বলে
CIRDAP-এর সদর দপ্তর	ঢাকা
ডিনামাইট আবিষ্কারক	আলফ্রেড নোবেল

ITTF বিশ্ব চ্যাম্পিয়নশিপ ২০২৫ পুরুষ এককে বিজয়ী কোন দেশের	চীন (ওয়াং চুকিন)
OSI মডেলে লেয়ার কয়টি	৭টি
"Terracotta" অর্থ	পোড়ামাটি (ইতালীয়: baked earth)
GSP+ সুবিধা দেয় কোন সংস্থা	ইউরোপীয় ইউনিয়ন (EU)

ব্যাংকিং ও আইসিটি — দ্রুত পুনরাবৃত্তি

প্রশ্ন	উত্তর
NPSB	National Payment Switch Bangladesh — বিভিন্ন ব্যাংকের ATM/POS/অনলাইন লেনদেন আন্তঃসংযোগ
BEFTN / RTGS / BACH	Bangladesh Electronic Funds Transfer Network (ব্যাচ) / Real Time Gross Settlement (তাৎক্ষণিক, একক) / Bangladesh Automated Clearing House
বাংলাদেশ ব্যাংকের গভর্নর পদ, সদর দপ্তর	সদর দপ্তর মতিঝিল, ঢাকা (সাম্প্রতিক গভর্নরের নাম পরীক্ষার আগে পত্রিকায় যাচাই করুন)
প্রথম বেসরকারি বাণিজ্যিক ব্যাংক	এবি ব্যাংক (১৯৮২)
বাংলাদেশে প্রথম কম্পিউটার	IBM 1620, পরমাণু শক্তি কেন্দ্র, ঢাকা (১৯৬৪)
প্রথম সাবমেরিন কেবল	SEA-ME-WE 4 (২০০৬, কক্সবাজার); দ্বিতীয় SEA-ME-WE 5 (কুয়াকাটা)
Bangabandhu Satellite-1	১২ মে ২০১৮
IMF / World Bank সদর দপ্তর	ওয়াশিংটন ডি.সি.
SAARC / BIMSTEC সদর দপ্তর	কাঠমান্ডু / ঢাকা
আন্তর্জাতিক সাক্ষরতা দিবস	৮ সেপ্টেম্বর

সাম্প্রতিক বিষয়

MCQ-র GK অংশে (১০) সাম্প্রতিক ঘটনা থাকতে পারে (যেমন D-তে ITTF ২০২৫)। পরীক্ষার আগের দিন ১৫ মিনিট পত্রিকার আন্তর্জাতিক ও অর্থনীতি পাতা দেখুন — এই বইয়ে সাম্প্রতিক সংখ্যা/নাম ইচ্ছা করে কম রাখা হয়েছে।

অধ্যায় ৮ — মডেল টেস্ট ও শেষ রাতের চেকলিস্ট

Two Full Mock Tests in the Job ID-25106 Pattern (MCQ 100 · Written 200)

> **কীভাবে দেবেন**
>
> আসল পরীক্ষার মতো সকাল ৯:৩০-এ শুরু করুন। MCQ ঠিক ৬০ মিনিট, তারপর লিখিত ১২০ মিনিট — মাঝে বিরতি নয়। ফোন দূরে, ক্যালকুলেটর নয়। শেষে MCQ-তে (সঠিক − ০.২৫ × ভুল) হিসাব করুন; লিখিত উত্তর নির্দেশিত অধ্যায়ের সমাধানের সাথে মেলান।

৮.১ মডেল টেস্ট ১

অংশ ক — MCQ (১০০ নম্বর, ৬০ মিনিট)

অধ্যায় ৬ থেকে নিচের ১০০টি প্রশ্ন একটানা করুন (উত্তর আগে ঢেকে রাখুন): **প্রশ্ন ১–৭৫** (Subjects Related ৭৫) + **প্রশ্ন ১১১–১২৫** (Mathematics ১৫) + **প্রশ্ন ১৩৬–১৪৫** (GK ১০)।

অংশ খ — লিখিত (২০০ নম্বর, ১২০ মিনিট)

Subjects Related (১৫ × ১০ = ১৫০) — বন্ধনীতে উত্তর কোথায় পাবেন

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. (অধ্যায় ৪ — hashing)
2. Write a C or Java program that prints all prime numbers between 1 and n, where n is given by the user. (অধ্যায় ৩)
3. Write a program to compute the GCD and LCM of two integers. (অধ্যায় ৩)
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. (অধ্যায় ৪)
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. (অধ্যায় ১ — class diagram-এর নিয়ম)
6. Differentiate method overloading and method overriding with a Java example. When does method hiding occur? (অধ্যায় ১, ৫)
7. Show the max-heap after inserting 15, 7, 20, 3, 18, 25 one by one, then after one extract-max. (অধ্যায় ৪)
8. Explain ACID properties with the example of a fund transfer between two accounts. (অধ্যায় ৫)
9. What is a deadlock? State the four necessary conditions and one prevention technique for each. (অধ্যায় ৫)
10. Compare TCP and UDP. Explain the TCP three-way handshake. (অধ্যায় ২, ৫)
11. Explain the CIA triad. Differentiate hashing and encryption with one algorithm example each. (অধ্যায় ২, ৫)
12. Critically compare the Waterfall model and Agile/Scrum. Which would you choose for a mobile banking app and why? (অধ্যায় ২, ৫)

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. (অধ্যায় ৪)
14. Prove that NAND is a universal gate. (অধ্যায় ২)
15. Explain how a queue can be implemented using two stacks. Give the time complexity of each operation. (অধ্যায় ২, ৪)

Mathematics (৪ × ৫ = ২০)

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 (৩০)

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.

উত্তর — মডেল টেস্ট ১

অংশ	উত্তর
IT ১ (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 ⇒ সূচক 0:44, 1:12, 2:13, 3:88, 4:23, 5:11, 6:94, 7:39, 9:20
IT ৭ (heap)	ইনসার্ট শেষে [25, 18, 20, 3, 7, 15]; extract-max ফেরত দেয় 25 ⇒ [20, 18, 15, 3, 7]
IT ১৩ (paging)	FIFO = ১০টি fault, LRU = ৯টি fault (ধাপের টেবিল অধ্যায় ৪-এর পদ্ধতিতে নিজে বানিয়ে মিলিয়ে নিন)
গণিত ১	1.40 × 0.85 = 1.19 ⇒ **19% লাভ**
গণিত ২	x + (x+2) + (x+4) = 78 ⇒ x = 24 ⇒ **24, 26, 28**
গণিত ৩	1/6 − 1/9 = 1/18 ⇒ **18 ঘণ্টা**
গণিত ৪	স্থির পানিতে (8 + 14)/2 = **11 km/h**; স্রোত (14 − 8)/2 = **3 km/h**
English ১	(a) is (b) would have helped (c) since (d) were (সদস্যরা আলাদা মত — plural) (e) to working (f) was (g) a unique (h) submit (i) do we see (j) composed of

৮.২ মডেল টেস্ট ২

অংশ ক — MCQ (১০০ নম্বর, ৬০ মিনিট)

অধ্যায় ৬ থেকে: **প্রশ্ন ৩৬–১১০** (Subjects Related ৭৫) + **প্রশ্ন ১২১–১৩৫** (Mathematics ১৫) + **প্রশ্ন ১৪১–১৫০** (GK ১০)। মডেল টেস্ট ১-এর সাথে কিছু প্রশ্ন মিলবে — সেগুলোতে আগের ভুল শুধরেছেন কি না, সেটাই দেখার বিষয়।

অংশ খ — লিখিত (২০০ নম্বর, ১২০ মিনিট)

Subjects Related (১৫ × ১০ = ১৫০)

1. Write a program to print Floyd's triangle (numbers) for n rows, and a second version that prints the 0/1 pattern. (অধ্যায় ১, ৩)
2. Write a function that returns the second largest element of an array in one pass. (অধ্যায় ৩)
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). (অধ্যায় ১, ৩)
4. Write SQL to find the second highest salary, and to find movies that belong to both 'Romance' and 'Comedy' genres. (অধ্যায় ১, ৪)
5. Explain normalization up to 3NF with an example table. (অধ্যায় ৫)
6. Represent a given directed graph using an adjacency list and an adjacency matrix. Which is better for a sparse graph, and why? (অধ্যায় ১, ৪)
7. Run Dijkstra's algorithm on a 5-node weighted graph and show the distance table after each step. (অধ্যায় ৪)
8. Describe the four pillars of OOP with a short Java example of each. (অধ্যায় ৫)
9. Explain the producer–consumer problem and its solution with semaphores. (অধ্যায় ২, ৫)
10. Describe a man-in-the-middle attack on Diffie–Hellman key exchange and how authentication prevents it. (অধ্যায় ১)
11. A new employee joins the bank. Describe secure user-account provisioning (password policy, MFA, least privilege, review, offboarding). (অধ্যায় ২, ৫)
12. Differentiate a process and a thread; give two advantages of multithreading. (অধ্যায় ২, ৫)
13. Explain IaaS, PaaS and SaaS with examples; differentiate a virtual machine and a container. (অধ্যায় ২, ৫)
14. Explain how OSPF builds its routing table. (অধ্যায় ২, ৫)
15. Compare supervised, unsupervised and reinforcement learning with one banking example each. (অধ্যায় ২, ৫)

Mathematics (৪ × ৫ = ২০)

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 (৩০)

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)

উত্তর — মডেল টেস্ট ২

অংশ	উত্তর
IT ৩ (f(5))	f(5): ছাপে 5 → f(3): 3 → f(1): 1 → f(−1) ফেরত → ছাপে 1 → 3 → 5 ⇒ **5 3 1 1 3 5**
গণিত ১	xy = (100 − 16)/4 = **21**; $x^2 + y^2$ = 100 − 42 = **58**
গণিত ২	0.15n = 45 ⇒ **n = 300**
গণিত ৩	(5/8)(4/7) = 20/56 = **5/14**
গণিত ৪	বাহু = 10 cm ⇒ ক্ষেত্রফল **100 cm^2**, পরিসীমা **40 cm**
English ১	(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

৮.৩ শেষ রাতের চেকলিস্ট (শুক্রবার রাত)

✓	বিষয়
☐	প্রবেশপত্র প্রিন্ট (১ কপি) — কিছু লিখবেন না; সঙ্গে শুধু কলম
☐	তেজগাঁও সরকারি উচ্চ বিদ্যালয়ে যাওয়ার পথ ও সময় ঠিক; ৮:৩০-এর মধ্যে পৌঁছানো (৯:১৫-এর পর প্রবেশ বন্ধ)
☐	ফোন, ঘড়ি, ক্যালকুলেটর, ব্যাগ, ব্যাংক কার্ড — কিছুই নয়
☐	শেষবার দেখুন: hash/heap/LRU trace পদ্ধতি · SQL JOIN + GROUP BY + HAVING · overloading/overriding/hiding · OSI + পোর্ট · ACID · deadlock ৪ শর্ত · UML class diagram চিহ্ন (inheritance ▷, composition ◆, aggregation ◇)
☐	English: conditionals, subjunctive (insist that + base), "one of the … who + plural verb", hardly/scarcely … when, no sooner … than
☐	নিজের ভুলের তালিকা একবার — তারপর ঘুম

> **হলে মনে রাখুন**
> কঠিন প্রশ্নে আটকে না থেকে পরেরটায় যান। প্রতিটি লিখিত উত্তরে অন্তত সংজ্ঞা + টেবিল/চিত্র + উদাহরণ দিন — পুরোটা না জানলেও আংশিক নম্বর আসে। MCQ-তে একটি অপশনও বাদ দিতে না পারলে খালি রাখুন।