Live MCQ™

৪৯তম স্পেশাল বিসিএস (শিক্ষা) বিষয়ভিত্তিক প্রস্তুতি

বিষয়: কম্পিউটার সায়েন্স (৯৭১)

Compendium PDF

(সিলেবাস অনুসারে সর্বাধিক গুরুত্বপূর্ণ টপিক ও এমসিকিউ-এর সমন্বয়ে রচিত)

Mentor:

আবু কাউছার

প্রভাষক (তথ্য ও যোগাযোগ প্রযুক্তি)

কালিয়াকৈর সরকারি কলেজ, কালিয়াকৈর, গাজীপুর

৩৮তম বিসিএস (সাধারণ শিক্ষা ক্যাডার)

বিষয় - কম্পিউটার সায়েন্স (সিএসই)

প্রিয় শিক্ষার্থীবৃন্দ, আসসালামু আলাইকুম। ৪৯তম (স্পেশাল) বিসিএস পরীক্ষা-২০২৫ এর বিষয়ভিত্তিক প্রস্তুতির কম্পিউটার সায়েন্স বিষয়ে PSC কর্তৃক নির্ধারিত সিলেবাসের ওপর সকল ক্লাস এবং পরীক্ষা ইতোমধ্যেই সম্পন্ন হয়েছে। আসন্ন চূড়ান্ত পরীক্ষাকে সামনে রেখে Live MCQ একাডেমিক টিম আপনাদের প্রস্তুতিতে শাণিত করতে বিশেষ কমপেনডিয়াম পিডিএফ প্রদান করছে। আশা করছি, এর মাধ্যমে আপনাদের প্রস্তুতি পূর্নাঙ্গ এবং ফলপ্রসু হবে।

সূচিপত্র

ক্রমিক	বিষয়
১।	পিএসসি কর্তৃক নির্ধারিত সিলেবাস ও মানবন্টন
২।	টপিক ভিত্তিক সাজেশন
৩।	টপিকের সংক্ষিপ্ত আলোচনা
৪।	শেষ মুহূর্তের নির্দেশনা
৫।	সর্বাধিক গুরুত্বপূর্ণ mcq প্রশ্ন উত্তর

Short Syllabus

Digital System: Number systems (binary, octal, hex, BCD, Code), Boolean algebra, Logic function Simplification, Kmap, QuineMcCluskey, Logic gates, Circuit design using Logic gates, Adders, Flip-flops, counters, and registers.

***Number systems (binary, octal, hex, BCD, Code)--বিভিন্ন কোড এবং সংখ্যা পদ্ধতির রুপান্তর (শর্ট কাট পদ্ধতি ব্যবহার করে সল্ভ করতে হবে)

***Boolean algebra-সূত্রাবলি থেকে ১ টি প্রশ্ন থাকবে
***Logic gates -- AND, OR, NOT, XOR, XNOR, NAND, NOR (Truth Table ভালো করে পড়লেই এখান থেকে যে কোন প্রশ্ন হোক না কেন সহজেই উত্তর করা যাবে)

**Logic function Simplification--Normal simplification (বিশেষ করে দ্যা মরগান থিওরেমের ব্যবহার)
**Kmap, QuineMcCluskey

***Adders --Half adder, Full adder (হাফ এডার দ্বারা ফুল এডার বাস্তবায়ন--ভালোভাবে বুজতে পারলেই এখান থেকে যে কোন প্রশ্ন সহজেই উত্তর করা যাবে)
***Flip-flops (D, T, JK Flip Flop), counters (Mod Counter দেখবে), and registers

Computer Programming: Introduction to programming, problem solving techniques, programming paradigms. Structured and object oriented programming, Language basics, program design, debugging. Programming in C/C++ and JAVA (overview).

***problem solving techniques-- ধাপগুলো ভালোভাবে পড়বে
***Structured and object-oriented programming -- পার্থক্য
***Programming in C/C++ and JAVA (overview).- অবজেক্ট ওরিয়েন্টেড প্রোগ্রামিং এর বৈশিষ্ট্য উদাহরণসহ পড়তে হবে
***variable declaration, data type, keyword (C)

Data Structures: Arrays, linked lists, stacks, queues, sparse/dense matrices. Recursion: applications. Tree structures: binary trees, BST, Graphs: representation, traversal. Sorting: merge, radix, quick, heap.

*** stacks, queues--(PUSH, POP, ENQUEUE, DEQUEUE)
*** Arrays (index এর ব্যবহার, মাল্টিডাইমেনশনাল এরে)
*** linked lists(single linked list, doubly linked list, circular linked list)
*** Recursion- ---example গুলো দেখবে
*** binary trees, BST বৈশিষ্ট্যগুলো দেখবে (root, height, level, parent, child)
***Sorting: merge, radix, quick, heap(sorting algorithm গুলোর টাইম কমপ্লেক্সিটি দেখবে। এখান থেকে প্রশ্ন হবার সম্ভাবনা অনেক বেশি)

Algorithm: Asymptotic notations, complexity analysis. Basic algorithms: divide & conquer, greedy, dynamic programming. Graph algorithms: shortest path, spanning trees. Approximation and parallel algorithms

***Asymptotic notations --(Big-Oh, Big-Omega, Big-Theta)
*** divide & conquer, greedy, dynamic programming (কোন কোন এলগরিদম এই টেকনিক ব্যবহার করে তাদের নাম আসার সম্ভাবনা অনেক বেশি)

***shortest path
***spanning trees

Computer Network & Internet: OSI & TCP/IP protocols. Network types (LAN, WAN), Ethernet, IP addressing, ICMP, ARP. Routing protocols, network security, wireless networks, DNS, Email, VPN, congestion control, ATM switches.

***OSI & TCP/IP protocols (লেয়ার গুলোর পার্থক্য ডিভাইস, প্রটোকল সহ পড়বে)
**Network types (LAN, WAN)
***IP addressing (IPv4, IPv6, Class A,B,C,D,E, Subnet Mask)
**Routing protocols
*** DNS
***Email, VPN
**network security, wireless networks

Data Communication: Transmission media: coaxial, twisted pair, optical. Analog/digital signals. Modulation: ASK, PSK, QAM, NRZ, Manchester. Multiplexing: TDM, FDM. Error detection, flow control. Circuit and packet switching.

***Transmission media: coaxial, twisted pair, optical (feature গুলো দেখবে)
***Modulation: ASK, PSK, QAM, NRZ, Manchester.(মূল ফাংশন দেখবে)
** Multiplexing: TDM, FDM (পার্থক্য দেখবে)
***Circuit and packet switching.(পার্থক্য দেখবে)

Database Management System: DBMS concepts, ER and relational models, normalization (1NF–BCNF), indexing techniques, query optimization. Transaction control, concurrency control. SQL queries and implementation.

***ER and relational models -- প্রতীক এবং কাজ
**** ACID properties
**** Key (Primary, Foreign, Composite, Candidate)
*** normalization (1NF–BCNF) (উদাহরণগুলো ভালো করে দেখবে)
*** SQL queries and implementation (w3schools.com থেকে প্র্যাক্টিস করবে। এখান থেকে একাধিক প্রশ্ন আসবে)

Discrete Mathematics: Sets, relations, propositional and predicate logic. Functions and recurrence relations. Counting principles. Graph theory and applications. Number theory: reversions, generating functions.

*** Sets, relations (সেট থিওরি যা ক্লাসে পড়ানো হয়েছে ভালো ভাবে দেখবে। এখান থেকে একাধিক প্রশ্ন হবার সম্ভাবনা বেশি)
*** propositional and predicate logic (উদাহরণ ভিত্তিক)
*** Functions (One-to-One, On-to function)
***Counting principles (normal permutation, combination থেকে প্রশ্ন হবে)

Numerical Analysis: Solving linear systems with Gaussian elimination and Gauss-Jordan elimination method. Interpolation: Newton's formula, Lagrange's formula. Numerical differentiations and integrations: Trapezoidal, Simpson's 1/3rd and 3/8th rule.

Operating System: OS overview and types, process and thread management. Scheduling algorithms. Inter-process communication, semaphores, deadlocks. Memory management: paging, segmentation. File management and security

*** OS overview and types, process and thread management (OS এর কার্যাবলি, প্রসেস এবং থ্রেডের পার্থক্য ভালো করে পড়বে)
*** deadlocks
*** Scheduling algorithms (FCFS, SJF, Round Robin, Priority Algorithm গ্যান্ট চার্ট , এভারেজ ওয়েটিং টাইম দ্রুততার সাথে বের করা শিখতে হবে)
*** Memory management: paging, segmentation (পেজিং এবং সেগমেন্টেশনের পার্থক্য)
***File management and security

Microprocessor & Interfacing: 8086 architecture, addressing modes, instruction set. Memory segmentation, interrupts, stack and TLB, memory mapped I/O. Bus interfacing. Interfacing with keyboard, monitor, I/O ports.

*** 8086 architecture, addressing modes, instruction set (BIU & EU এর কার্যাবলি ভালো করে পড়বে। জেনারেল পারপাস রেজিস্টারগুলোর কাজ অতি সংক্ষেপে পড়বে। ফ্ল্যাগ রেজিস্টার এবং ALU এর ফাংশনালিটি থেকে প্রশ্ন হবে)

*** memory mapped I/O
**memory mapped I/O.
***Bus interfacing

Computer Organization & Architecture: Fundamentals of computer design. ALU and control design. Instruction cycle, pipelining, hazards. Cache memory, memory hierarchy. Systolic arrays, Fault tolerance. Bus and microprogrammed control.

Compiler and theory of computation: Introduction to compilary. Basic issues, logical analysis, lexical analysis, syntax analyses. Semantic analysis, type cheeking, run-time environments, code generation, code optimization and language theory.

Artificial Intelligence: Overview of AI. General concepts of knowledge. Introduction to PROLOG. Knowledge representation. Intelligent agents. First order logic. Knowledge organization, Natural language processing, expert systems and computer vision.

MCQ solve করার টেকনিক

1. **প্রশ্ন মনোযোগ দিয়ে পড়তে হবে**
2. **সব অপশন যাচাই করবে এবং Elimination Method Apply করবে**

   →ট্রিক অপশন থাকতে পারে

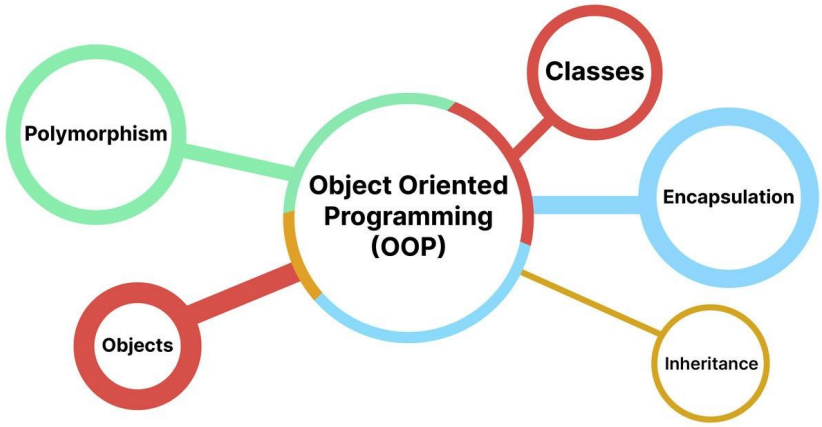
3. **শুরুতেই** শতভাগ নিশ্চিত **প্রশ্নের উত্তর দিতে হবে এবং** বিভ্রান্তিমূলক প্রশ্ন এড়িয়ে **যেতে হবে**
4. **দ্বিতীয় ভাগে:** ৫০-৫০ সম্ভাবনাময় উত্তরের **ক্ষেত্রে** ভ্যালু ইনপুট চেকিং/লজিক **দিয়ে সঠিক অপশন বেছে নিতে হবে।**
5. **সময়ের দিকে** বারংবার খেয়াল **রাখতে হবে।**

বিষয় ভিত্তিক গুরুত্বপূর্ণ টপিক সমূহ

Programming

Object Oriented Programming

Object-Oriented Programming (OOP) is a coding approach that organizes programs into objects, combining data and behavior to model real-world entities efficiently.



Core Principles:

- **Encapsulation:** Data ও methods একত্রিত করে access restrict করা।
- **Inheritance:** One class থেকে another class-এ properties পাওয়া।
- **Polymorphism:** Same interface for different data types
- **Abstraction:** Complex details hide করে simple interface

Aspect	Structured Programming	OOP
Definition	Code is organized in procedures or functions with a linear, step-by-step flow.	Code is organized around objects that combine data and behavior.
Paradigm	Procedural, focuses on functions.	Object-based, focuses on objects and classes.
Key Components	Functions, loops, conditionals (if-else).	Classes, objects, methods, attributes.
Data Handling	Data and functions are separate.	Data and methods are encapsulated within objects.
Abstraction	Limited, achieved through functions.	Strong, achieved through classes and interfaces.
Inheritance	Not supported .	Supported, allows code reuse via parent-child classes.
Polymorphism	Not supported or limited (e.g., function overloading).	Supported via method overriding/overloading.
Encapsulation	Weak, data is often globally accessible.	Strong, data is private and accessed via methods.
Code Reusability	Limited, achieved through functions.	High, through inheritance and polymorphism.
Scalability	Suitable for small programs, complex for large systems	Suitable for large, complex systems.
Example Languages	C, Pascal	Java, C++, Python
Use Cases	Simple scripts, system programming.	Enterprise apps, web development, large systems.

Program Design

i. Requirement Analysis (প্রয়োজনীয়তা বিশ্লেষণ)
ii. Algorithm Development (অ্যালগরিদম তৈরি)
iii. Modular Design (মডুলার ডিজাইন)
iv. Data Structure Selection (ডেটা স্ট্রাকচার নির্বাচন)
v. Coding/Implementation (কোডিং/বাস্তবায়ন)
vi. Testing Plan (পরীক্ষার পরিকল্পনা)
vii. Debugging (ডিবাগিং)
viii. Optimization (অপ্টিমাইজেশন)
ix. Documentation (ডকুমেন্টেশন)
x. Deployment (ডিপ্লয়মেন্ট)
xi. Maintenance (রক্ষণাবেক্ষণ)

Difference between C, C++, Java:

Feature	C	C++	Java
Paradigm	Procedural	Procedural, OOP	OOP
Memory Management	Manual (pointers)	Manual (pointers)	Automatic (GC)
Platform	Platform-dependent	Platform-dependent	Platform-independent
Performance	Very fast	Fast	Slower (JVM)
Use Cases	OS, embedded systems	Games, systems	Web, mobile, enterprise

Algorithm

Table: Asymptotic Notation

Notation	Definition	Bound Type	Interpretation	Example
$O(f(n))$	$g(n) \leq c \cdot f(n)$ for some constant $c > 0$ and $n \geq n_0$	Upper Bound	Worst-case time complexity; function grows no faster than f(n).	$O(n^2)$ for bubble sort
$\Theta(f(n))$	$c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$ for constants $c_1, c_2 > 0$ and $n \geq n_0$	Tight Bound	Function grows exactly at the rate of f(n).	$\Theta(n \log n)$ for merge sort
$\Omega(f(n))$	$g(n) \geq c \cdot f(n)$ for some constant $c > 0$ and $n \geq n_0$	Lower Bound	Best-case time complexity; function grows at least as fast as f(n).	$\Omega(1)$ for constant-time operations

Sorting Algorithms

Algorithm	Best Case	Average Case	Worst Case	Space Complexity	Stable?	Notes
Bubble Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)	Yes	Repeatedly swaps adjacent elements if out of order. Inefficient for large datasets.
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	O(1)	No	Finds min/max in unsorted portion and places it at the end.
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$	O(1)	Yes	Builds sorted array one element at a time; efficient for small or nearly sorted data.
Merge Sort	O(n log n)	O(n log n)	O(n log n)	O(n)	Yes	Divide-and-conquer; splits array, sorts, and merges. Stable and predictable.
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$	O(log n)	No	Divide-and-conquer; uses pivot. Worst case rare with good pivot selection.
Heap Sort	O(n log n)	O(n log n)	O(n log n)	O(1)	No	Uses max/min heap to sort. Not stable but in-place.
Counting Sort	O(n + k)	O(n + k)	O(n + k)	O(k)	Yes	Non-comparison-based; uses range of keys (k). Efficient for small key ranges.
Radix Sort	O(nk)	O(nk)	O(nk)	O(n + k)	Yes	Non-comparison-based; sorts by digits. k is max digits or key size.
Bucket Sort	O(n + k)	O(n + k)	$O(n^2)$	O(n + k)	Yes	Distributes elements into buckets, sorts buckets. Best for uniformly distributed data.

Searching Algorithms

Algorithm	Best Case	Average Case	Worst Case	Space Complexity	Notes
Linear Search	O(1)	O(n)	O(n)	O(1)	Sequentially checks each element. Works on unsorted data.
Binary Search	O(1)	O(log n)	O(log n)	O(1) (iterative), O(log n) (recursive)	Requires sorted data. Divides search space in half each step.
Jump Search	O(1)	O(√n)	O(√n)	O(1)	Jumps fixed steps, then linear search. Works on sorted arrays.
Exponential Search	O(1)	O(log n)	O(log n)	O(1)	Finds range using exponential steps, then binary search. Sorted arrays.
Interpolation Search	O(1)	O(log log n)	O(n)	O(1)	Assumes uniform distribution; estimates position. Sorted arrays.
Hash-based Search	O(1)	O(1)	O(n)	O(n)	Uses hash table. Worst case occurs with collisions.

Dynamic Programming vs Greedy vs Divide & Conquer

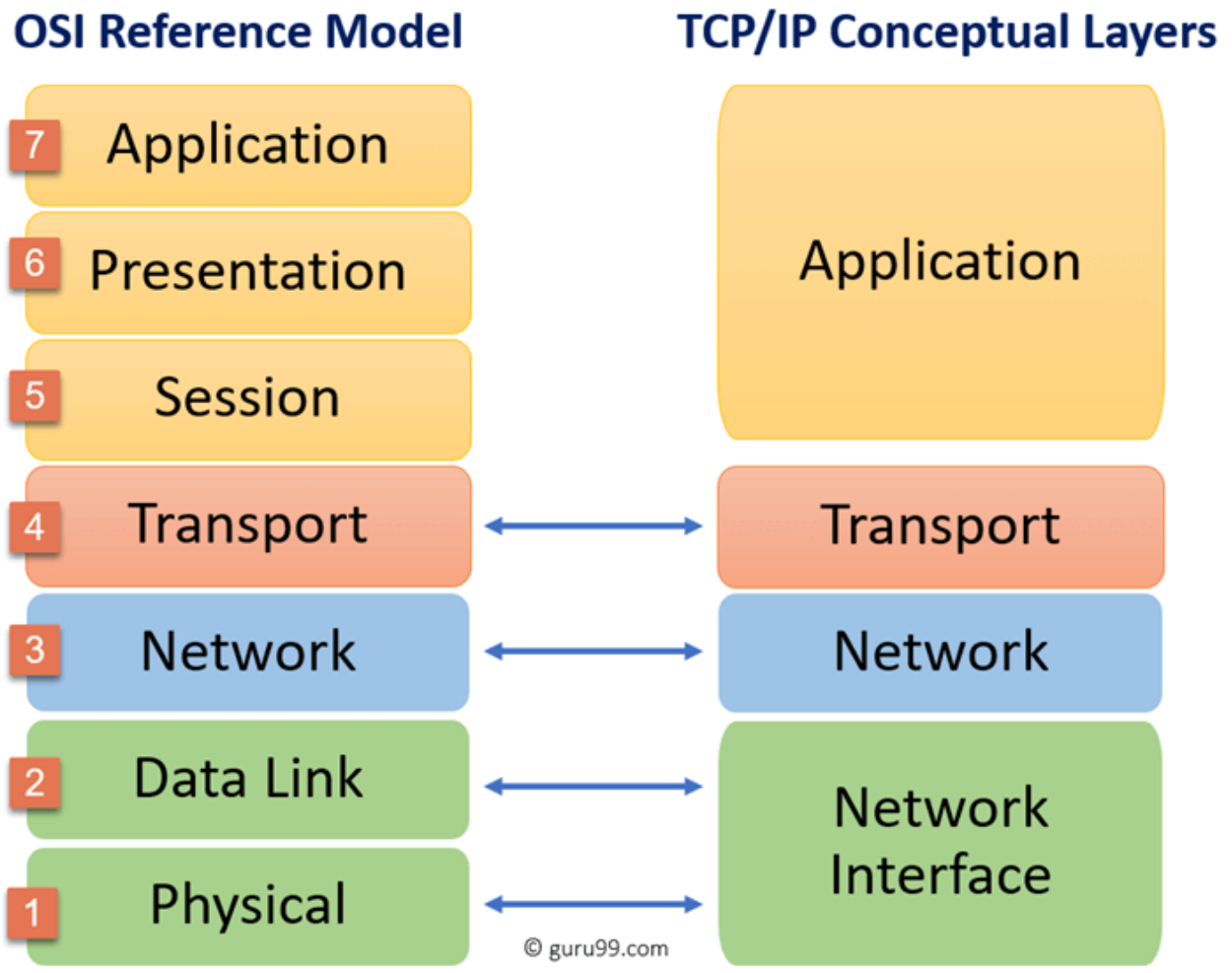
Aspect	Dynamic Programming	Greedy	Divide and Conquer
Definition	Solves problems by breaking them into overlapping subproblems, solving each subproblem once, and storing results for reuse.	Makes locally optimal choices at each step, hoping to find a global optimum.	Divides problem into independent subproblems, solves them recursively, and combines results.
Approach	Bottom-up (iterative) or top-down (recursive with memoization). Solves subproblems and builds solution using stored results.	Makes the best choice at each step without reconsidering past choices.	Divides problem into smaller, non-overlapping subproblems, solves them, and merges results.
Subproblem Dependency	Subproblems are overlapping; solutions are reused.	No subproblem dependency; each choice is independent.	Subproblems are independent; no reuse of solutions.

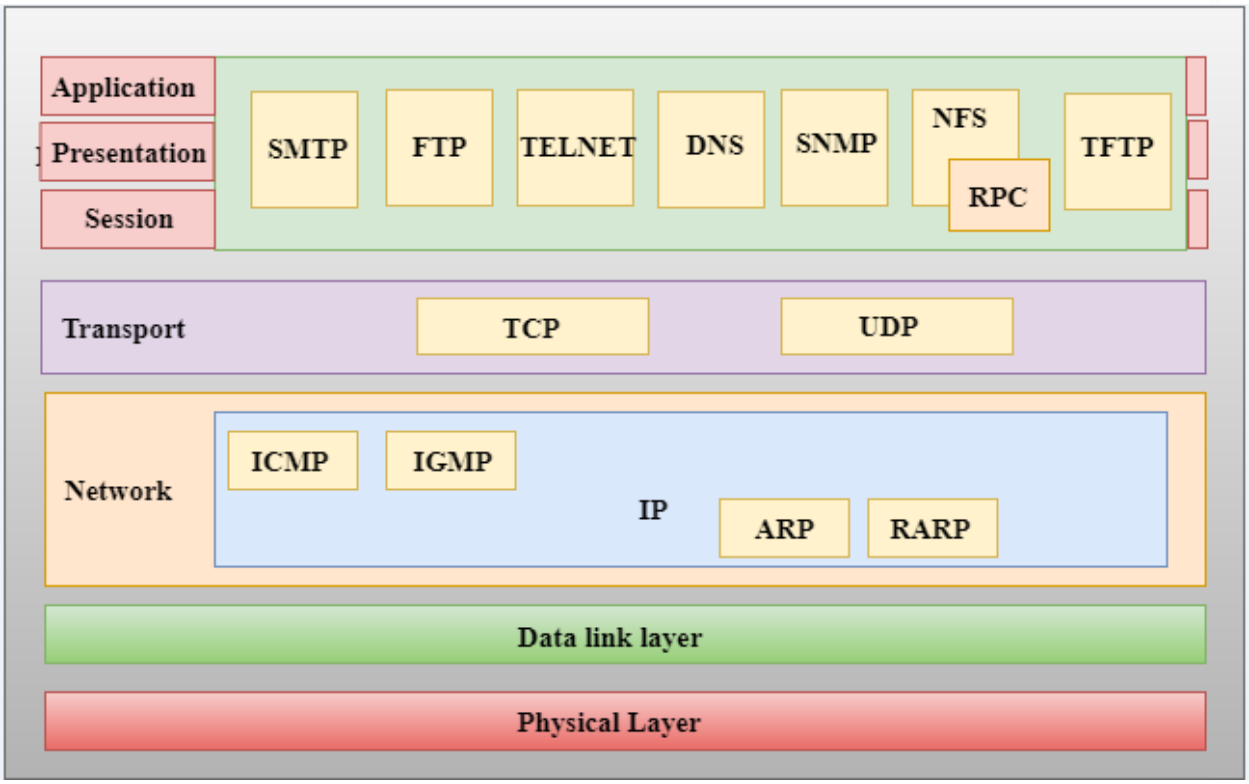
Optimality	Guarantees optimal solution for problems with optimal substructure and overlapping subproblems.	May not guarantee optimal solution; works for problems with greedy choice property (e.g., matroid problems).	Guarantees correct solution but not necessarily optimal unless designed for optimization.
Time Complexity	Typically polynomial (e.g., $O(n^2)$, $O(nk)$) depending on problem and subproblem count.	Often linear or near-linear (e.g., $O(n)$, $O(n \log n)$) due to single-pass decisions.	Varies (e.g., $O(n \log n)$ for merge sort, $O(2^n)$ for naive recursive algorithms).
Space Complexity	$O(n)$ to $O(n^2)$ for storing subproblem solutions (can be optimized in some cases).	Usually $O(1)$ or $O(n)$ as it avoids storing subproblem results.	$O(\log n)$ for recursive call stack, or more if storing intermediate results (e.g., $O(n)$ for merge sort).
Examples	- Fibonacci (memoized) - Knapsack (0/1) - Longest Common Subsequence - Matrix Chain Multiplication	- Kruskal's Algorithm - Dijkstra's Algorithm - Huffman Coding - Fractional Knapsack	- Merge Sort - Quick Sort - Binary Search - Strassen's Matrix Multiplication
When to Use	Problems with overlapping subproblems and optimal substructure (e.g., optimization problems like shortest paths).	Problems where local optimal choices lead to global optimum (e.g., minimum spanning tree).	Problems that can be split into independent subproblems (e.g., sorting, searching).
Advantages	- Guarantees optimality - Reduces computation for overlapping subproblems	- Simple and fast - Low memory usage	- Effective for parallelization - Clear recursive structure
Disadvantages	- Higher memory usage - Can be complex to design	- May not yield optimal solution - Limited applicability	- Recursive overhead - May require significant merging effort
Stability	Stable; always produces consistent results for same input.	Stable for specific problems; depends on greedy choice logic.	Stable if designed to be (e.g., merge sort is stable, quick sort is not).

Computer Networks & Internet

OSI (Open Source Interconnection) 7 Layer Model

Layer	Application/Example		Central Device/ Protocols		DOD4 Model
Application (7) Serves as the window for users and application processes to access the network services.	**End User layer** Program that opens what was sent or creates what is to be sent Resource sharing • Remote file access • Remote printer access • Directory services • Network management		**User Applications** SMTP	GATEWAY	Process
Presentation (6) Formats the data to be presented to the Application layer. It can be viewed as the "Translator" for the network.	**Syntax layer** encrypt & decrypt (if needed) Character code translation • Data conversion • Data compression • Data encryption • **Character Set Translation**		JPEG/ASCII EBDIC/TIFF/GIF PICT		
Session (5) Allows session establishment between processes running on different stations.	**Synch & send to ports** (logical ports) Session establishment, maintenance and termination • Session support - perform security, name recognition, logging, etc.		**Logical Ports** RPC/SQL/NFS NetBIOS names		
Transport (4) Ensures that messages are delivered error-free, in sequence, and with no losses or duplications.	**TCP** Host to Host, Flow Control Message segmentation • Message acknowledgement • Message traffic control • Session multiplexing	PACKET FILTERING	TCP/SPX/UDP		Host to Host
Network (3) Controls the operations of the subnet, deciding which physical path the data takes.	**Packets** ("letter", contains IP address) Routing • Subnet traffic control • Frame fragmentation • Logical-physical address mapping • Subnet usage accounting		**Routers** IP/IPX/ICMP	Can be used on all layers	Internet
Data Link (2) Provides error-free transfer of data frames from one node to another over the Physical layer.	**Frames** ("envelopes", contains MAC address) [NIC card — Switch — NIC card] (end to end) Establishes & terminates the logical link between nodes • Frame traffic control • Frame sequencing • Frame acknowledgment • Frame delimiting • Frame error checking • Media access control		**Switch** **Bridge** **WAP** PPP/SLIP	Land Based Layers	Network
Physical (1) Concerned with the transmission and reception of the unstructured raw bit stream over the physical medium.	**Physical structure** Cables, hubs, etc. Data Encoding • Physical medium attachment • Transmission technique - Baseband or Broadband • Physical medium transmission Bits & Volts		**Hub**		





Protocol & Their functions

Protocol	Layer (OSI/TCP/IP)	Function	How It Works (In Short)
HTTP	Application (Application)	Transfers web content	Facilitates client-server communication (e.g., browsers request web pages from servers) using request-response messages over TCP.
HTTPS	Application (Application)	Secure web content transfer	HTTP over SSL/TLS, encrypting data for secure communication between client and server.
FTP	Application (Application)	File transfer	Enables uploading/downloading files between client and server using TCP; supports authentication.
SMTP	Application (Application)	Email sending	Transfers outgoing emails from client to server or between servers using TCP; handles message routing.
POP3	Application (Application)	Email retrieval	Retrieves emails from a server to a client over TCP; typically deletes emails from server after download.
IMAP	Application (Application)	Email access	Allows clients to access and manage emails on a server over TCP; supports synchronization across devices.
DNS	Application (Application)	Domain name resolution	Translates domain names (e.g., google.com) to IP addresses using a distributed database over UDP/TCP.
TCP	Transport (Transport)	Reliable data transfer	Establishes connection-oriented, error-checked, ordered delivery of data packets between devices.
UDP	Transport (Transport)	Fast, unreliable data transfer	Sends datagrams without connection setup; used for low-latency applications like streaming.
IP (IPv4/IPv6)	Network (Internet)	Addressing and routing	Assigns IP addresses and routes packets across networks; IPv6 supports larger address space.
ICMP	Network (Internet)	Network diagnostics	Sends error messages and diagnostic data (e.g., ping) to manage network issues; operates over IP.
ARP	Link (Network Interface)	IP-to-MAC address mapping	Resolves IP addresses to MAC addresses within a local network for packet delivery.
DHCP	Application (Application)	Dynamic IP allocation	Automatically assigns IP addresses and network configurations to devices in a network over UDP.
BGP	Application (Application)	Inter-domain routing	Manages routing between autonomous systems (e.g., ISPs) using path vector routing

			over TCP.
RIP	Application (Application)	Intra-domain routing	Distance-vector routing protocol; shares routing tables with neighbors to determine paths; uses UDP.
OSPF	Application (Application)	Intra-domain routing	Link-state routing protocol; calculates shortest paths within a network using Dijkstra's algorithm; uses IP directly.
SNMP	Application (Application)	Network management	Monitors and manages network devices; collects device status and performance data over UDP.
Telnet	Application (Application)	Remote terminal access	Provides text-based remote access to devices for configuration and management over TCP; unsecured.
SSH	Application (Application)	Secure remote access	Encrypts remote terminal sessions for secure device management and file transfer over TCP.

Network Security

Goal: Ensure confidentiality, integrity, and availability (CIA) of network information.

- **Firewall: Device or software that filters traffic based on rules, blocking unauthorized access (e.g., packet filtering, stateful inspection).**
- **VPN (Virtual Private Network): Encrypts data over public networks, ensuring secure remote access using protocols like IPsec or OpenVPN.**
- **IDS/IPS: Intrusion Detection/Prevention Systems monitor and block suspicious network activity (e.g., Snort, Cisco IPS).**
- **Encryption Protocols: HTTPS (SSL/TLS) for secure web traffic; WPA3 for Wi-Fi security.**
- **Authentication: Verifies user/device identity using passwords, biometrics, or multi-factor authentication (MFA).**
- **Common Threats: Phishing (fake emails), DDoS (traffic overload), Man-in-the-Middle (data interception), malware.**

VPN

- **Definition: A VPN creates a secure, encrypted connection over the internet, protecting data and privacy.**
- **Purpose: Enables secure remote access to networks, bypassing geo-restrictions and ensuring anonymity.**
- **Encryption: Uses protocols like OpenVPN, L2TP/IPSec, or WireGuard to encrypt data, preventing interception.**
- **IP Masking: Hides the user's real IP address, showing the VPN server's IP instead for privacy.**
- **Tunneling: Data is transmitted through a secure "tunnel" between the user's device and the VPN server.**

- Applications: Used for secure browsing, accessing restricted content, remote work, and protecting data on public Wi-Fi.
- Protocols: Common protocols include OpenVPN (secure, open-source), PPTP (fast but less secure), and IKEv2 (mobile-friendly).
- Security Benefits: Protects against hacking, surveillance, and data theft; essential for sensitive transactions.
- Limitations: May slow down internet speed due to encryption; some services block VPNs.
- Use in Bangladesh: Helps access blocked sites, secure online banking, and protect privacy on public networks.

E-mail:

- Email Definition: Electronic mail enables sending digital messages over networks, using protocols like SMTP, POP3, or IMAP.
- CC (Carbon Copy): Recipients added in the CC field receive a copy of the email; visible to all recipients; used for transparency or inclusion.
- BCC (Blind Carbon Copy): Recipients in the BCC field receive the email but are hidden from other recipients; ensures privacy.
- To Field: Primary recipient(s) of the email; their address is visible to all; used for direct communication.
- Subject Line: Brief summary of the email's purpose; helps recipients understand the email's context before opening.
- Email Body: Main content of the email, containing the message, which can include text, images, or attachments.
- Attachments: Files (e.g., PDFs, images) sent with the email; size limits vary by email service.
- Email Protocols: SMTP (sending emails), POP3 (retrieving, deletes from server), IMAP (retrieving, syncs with server).
- Security: Encryption (e.g., TLS) protects emails; phishing and spam are common risks; use strong passwords.
- Etiquette: Use clear, professional language; avoid overusing CC/BCC; include signature for formal emails.

Data Communication

Table: Differences Between Twisted Pair, Coaxial Cable, and Fiber Optic

Aspect	Twisted Pair	Coaxial Cable	Fiber Optic
Structure	Two insulated copper wires twisted together (e.g., UTP, STP).	Central conductor surrounded by a shield, insulator, and outer conductor.	Glass or plastic core that transmits light, surrounded by cladding and protective coating.
Data Transmission	Electrical signals.	Electrical signals.	Light signals (optical).
Types	Unshielded Twisted Pair (UTP), Shielded Twisted Pair (STP).	RG-6, RG-59, etc. (based on impedance and use).	Single-mode, Multi-mode.
Bandwidth	Up to 10 Gbps (e.g., Cat6, Cat7).	Up to 1 Gbps (modern variants).	Up to 100 Gbps or more (single-mode).
Distance	~100 meters (signal degrades beyond).	~500 meters (varies by type).	Up to 40+ km (single-mode) without repeaters.
Cost	Low (cheapest among the three).	Moderate.	High (expensive installation and equipment).
Interference	Susceptible to electromagnetic interference (EMI); STP less so than UTP.	Moderate resistance to EMI due to shielding.	Immune to EMI; no electrical interference.
Installation	Easy to install and terminate; flexible.	Moderately easy; less flexible than twisted pair.	Complex; requires specialized equipment and skills.
Durability	Moderate; prone to physical damage.	Good; more robust than twisted pair.	High; resistant to environmental factors but fragile if bent sharply.
Security	Vulnerable to tapping.	Moderately secure; harder to tap than twisted pair.	Highly secure; difficult to tap without detection.
Applications	Ethernet networks (e.g., LANs, Cat5e/Cat6 for offices).	Cable TV, broadband internet, CCTV.	High-speed networks, long-distance telecom, data centers.
Weight	Lightweight.	Heavier than twisted pair.	Lightest; thin and flexible.
Power Consumption	Supports Power over Ethernet (PoE).	Can carry power (e.g., in cable TV systems).	No power transmission; requires separate power for devices.
Attenuation	High; signal loss over short distances.	Moderate; less loss than twisted pair.	Very low; minimal signal loss over long distances.

Table: Differences Between ASK, FSK, and PSK

Aspect	ASK (Amplitude Shift Keying)	FSK (Frequency Shift Keying)	PSK (Phase Shift Keying)
Definition	Modulates the amplitude of the carrier signal to represent digital data (e.g., 0 or 1).	Modulates the frequency of the carrier signal to represent digital data.	Modulates the phase of the carrier signal to represent digital data.
How It Works	Changes carrier signal amplitude (e.g., high amplitude for 1, low for 0) while keeping frequency and phase constant.	Changes carrier signal frequency (e.g., high frequency for 1, low for 0) while keeping amplitude constant.	Changes carrier signal phase (e.g., 0° for 0, 180° for 1 in BPSK) while keeping amplitude and frequency constant.
Types/Variants	- Binary ASK (BASK) - Multilevel ASK (e.g., 4-ASK)	- Binary FSK (BFSK) - Continuous Phase FSK (CPFSK) - Minimum Shift Keying (MSK)	- Binary PSK (BPSK) - Quadrature PSK (QPSK) - 8-PSK, 16-PSK, etc.
Bandwidth Efficiency	Low to moderate; higher levels (e.g., 4-ASK) improve efficiency but increase complexity.	Lower than PSK; requires more bandwidth for same data rate.	High; PSK (e.g., QPSK) encodes more bits per symbol, reducing bandwidth needs.
Noise Immunity	Poor; highly susceptible to noise and amplitude variations.	Moderate; less affected by amplitude noise but sensitive to frequency interference.	High; robust against noise, especially in higher-order PSK (e.g., QPSK).
Power Efficiency	Less efficient; requires distinct amplitude levels, which can be power-intensive.	Moderate; constant amplitude reduces power needs compared to ASK.	High; constant amplitude (e.g., in BPSK/QPSK) optimizes power usage.
Complexity	Simple to implement; basic circuitry.	Moderate; requires frequency synthesis and detection.	Complex; needs precise phase synchronization and coherent detection.
Advantages	- Simple and low-cost implementation. - Suitable for basic applications.	- Resistant to amplitude noise. - Reliable in environments with varying signal strength.	- High bandwidth efficiency. - Robust against noise; supports high data rates.

Disadvantages	- Highly sensitive to noise and interference. - Limited data rate due to amplitude constraints.	- Higher bandwidth requirement. - Susceptible to frequency-specific interference.	- Complex implementation. - Requires accurate phase synchronization.
Applications	- Optical fiber communication. - Simple RF systems (e.g., remote controls).	- Low-speed modems. - RFID, wireless sensor networks. - GSM (e.g., GMSK variant).	- High-speed communication (e.g., Wi-Fi, satellite, 4G/5G). - QPSK in digital TV and modems.
Bit Error Rate (BER)	Higher BER in noisy environments.	Moderate BER; better than ASK in noisy conditions.	Lower BER; performs best in noisy environments.
Example	1: High amplitude, 0: Low amplitude.	1: 1200 Hz, 0: 2200 Hz (e.g., early modems).	BPSK: 0° for 0, 180° for 1; QPSK: 4 phase states for 2 bits.

Error Detection & Flow Control

Types of Errors

There may be three types of errors:

- **Single bit error**

In a frame, there is only one bit, anywhere though, which is corrupt.

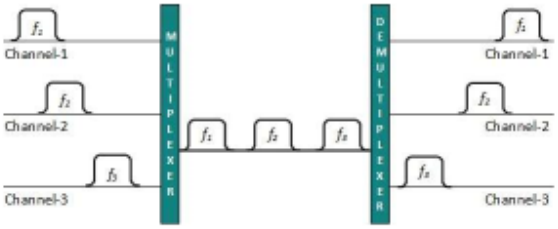
- **Multiple bits error**

Frame is received with more than one bits in corrupted state.

- **Burst error**

Frame contains more than1 consecutive bits corrupted.

FDM



Divides a communication channel into multiple frequency bands, each carrying a separate signal. FDM is an analog technology.

- **How it works**: Each signal is assigned a unique frequency range within the shared medium.

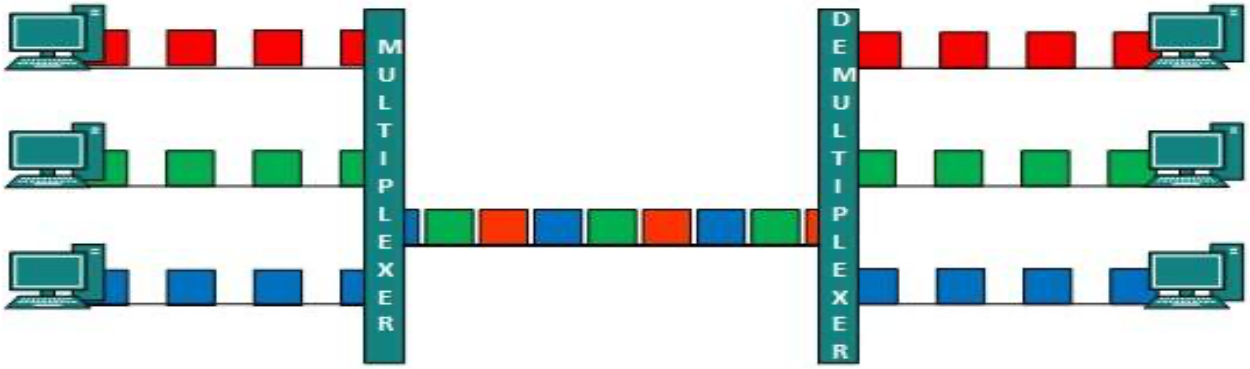


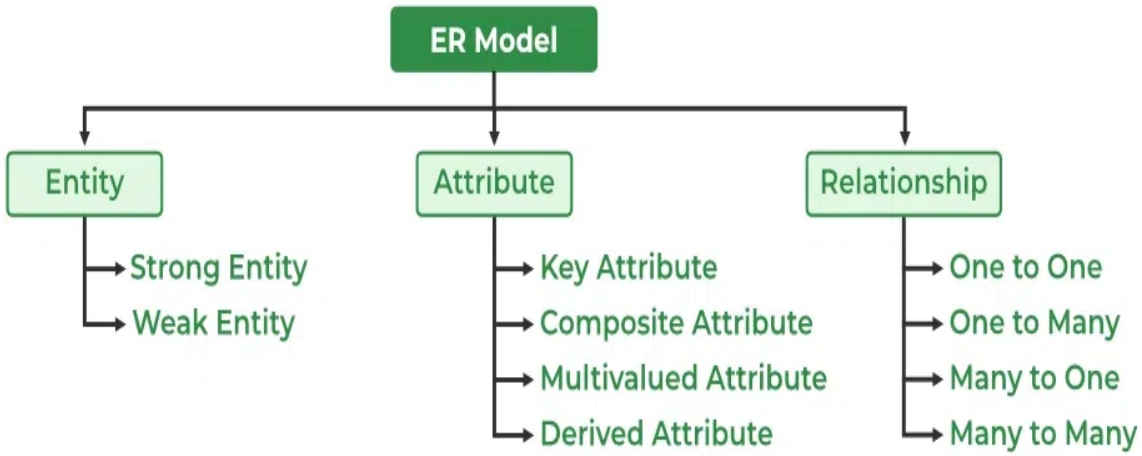
Fig: TDM

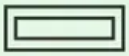
Table: Differences Between Circuit Switching and Packet Switching

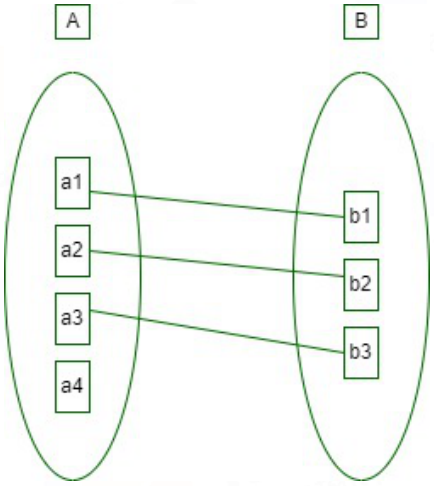
Aspect	Circuit Switching	Packet Switching
Definition	Establishes a dedicated communication path between two nodes for the entire duration of the session.	Breaks data into packets, which are sent independently over the network and reassembled at the destination.
How It Works	A fixed path (circuit) is reserved before data transfer, used exclusively until the session ends.	Data is divided into small packets, each routed individually based on network conditions, reassembled at the destination.
Connection Type	Connection-oriented; requires setup and teardown phases.	Connectionless (e.g., IP) or connection-oriented (e.g., TCP); no dedicated path.
Resource Utilization	Inefficient; resources are reserved even when no data is sent (e.g., during silence in a call).	Efficient; resources are shared dynamically, used only when packets are transmitted.
Bandwidth Allocation	Fixed bandwidth allocated for the entire session.	Dynamic bandwidth; packets share available network resources.

Delay	Low and consistent delay once the circuit is established.	Variable delay due to packet queuing and routing.
Reliability	Reliable for duration of session; no packet loss if circuit is maintained.	Less reliable; packets may be lost or arrive out of order, requiring retransmission or reordering.
Error Handling	Minimal; errors handled at physical layer or by retransmission after circuit failure.	Built-in; protocols (e.g., TCP) handle error detection, retransmission, and reordering.
Setup Time	High; requires time to establish and tear down the circuit.	Low or none; packets are sent immediately without dedicated setup.
Data Transmission	Continuous; ideal for steady, real-time data streams.	Burst; suitable for variable, non-continuous data transfer.
Advantages	- Predictable performance (low jitter). - Ideal for real-time applications like voice calls.	- Efficient use of network resources. - Scalable for large networks like the Internet.
Disadvantages	- Inefficient for bursty data. - Wastes resources during idle periods.	- Variable delay and potential packet loss. - Complex routing and reassembly.
Applications	- Traditional telephone networks (PSTN). - Real-time voice and video calls (e.g., ISDN).	- Internet (IP networks). - Email, web browsing, file transfers (e.g., TCP/IP).
Examples	- Analog telephone calls. - Dedicated leased lines.	- Ethernet, Internet (TCP/IP), VoIP.
Scalability	Limited; fixed paths scale poorly for large networks.	High; flexible routing supports large, dynamic networks.
Cost	Higher; dedicated resources increase costs.	Lower; shared resources reduce costs.

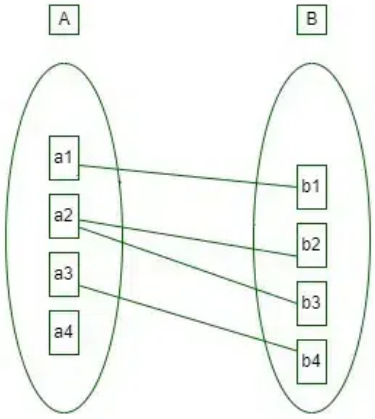
DBMS



Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity



Set Representation of One-to-One



Set Representation of One-to-Many

Normalization : Contains repeating groups and non-atomic values.

StudentID	StudentName	Course	Instructor	InstructorOffice
S1	Alice	Math, CS	ProfX, ProfY	OfficeA, OfficeB
S2	Bob	CS	ProfY	OfficeB

1NF (First Normal Form) Eliminate repeating groups; ensure atomic values.

StudentID	StudentName	Course	Instructor	InstructorOffice
S1	Alice	Math	ProfX	OfficeA
S1	Alice	CS	ProfY	OfficeB
S2	Bob	CS	ProfY	OfficeB

2NF (Second Normal Form) Must be in 1NF; eliminate partial dependencies (non-key attributes depend on entire primary key: StudentID, Course).**Student**:

StudentID	StudentName
S1	Alice
S2	Bob

Enrollment:

StudentID	Course	Instructor	InstructorOffice
S1	Math	ProfX	OfficeA
S1	CS	ProfY	OfficeB
S2	CS	ProfY	OfficeB

Enrollment:

StudentID	Course	Instructor	InstructorOffice
S1	Math	ProfX	OfficeA
S1	CS	ProfY	OfficeB
S2	CS	ProfY	OfficeB

3NF (Third Normal Form) Must be in 2NF; eliminate transitive dependencies (non-key attributes depend only on primary key).

Student:

StudentID	StudentName
S1	Alice
S2	Bob

Instructor:

Instructor	InstructorOffice
ProfX	OfficeA
ProfY	OfficeB

Enrollment:

StudentID	Course	Instructor
S1	Math	ProfX
S1	CS	ProfY
S2	CS	ProfY

BCNF (Boyce-Codd Normal Form) Must be in 3NF; for every dependency (X → Y), X must be a superkey (e.g., Instructor → Course).

Student:

StudentID	StudentName
S1	Alice
S2	Bob

Instructor:

Instructor	InstructorOffice
ProfX	OfficeA
ProfY	OfficeB

CourseInstructor:

Instructor	Course
ProfX	Math
ProfY	CS

StudentCourse:

StudentID	Course
S1	Math
S1	CS
S2	CS

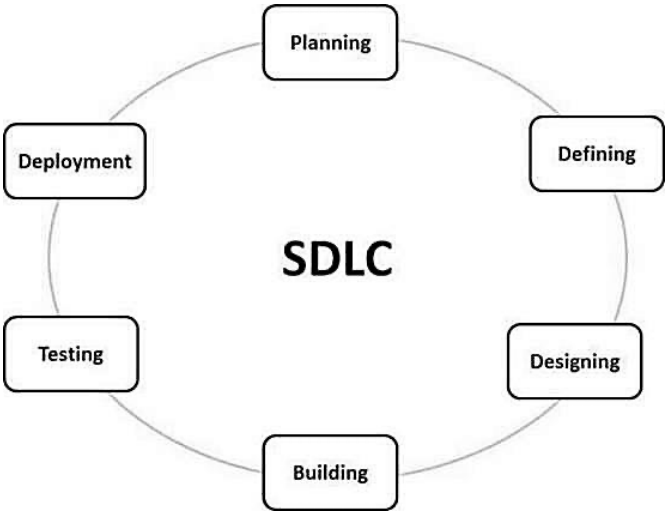
Table: Common SQL Queries and Their Functions

SQL Query	Purpose	How It Works	Example
SELECT	Retrieves data from one or more tables.	Queries specified columns or all columns (*) from a table, optionally with conditions (WHERE), sorting (ORDER BY), or grouping (GROUP BY).	SELECT name, age FROM users WHERE age > 18 ORDER BY name; Retrieves names and ages of users over 18, sorted by name.
INSERT	Adds new rows to a table.	Specifies the table and values to insert, either for all columns or specific ones. Can insert multiple rows at once.	INSERT INTO users (name, age) VALUES ('Alice', 25); Adds a new user named Alice, aged 25.
UPDATE	Modifies existing data in a table.	Updates specified columns in rows matching a condition (WHERE). Without WHERE, all rows are updated.	UPDATE users SET age = 26 WHERE name = 'Alice'; Changes Alice's age to 26.

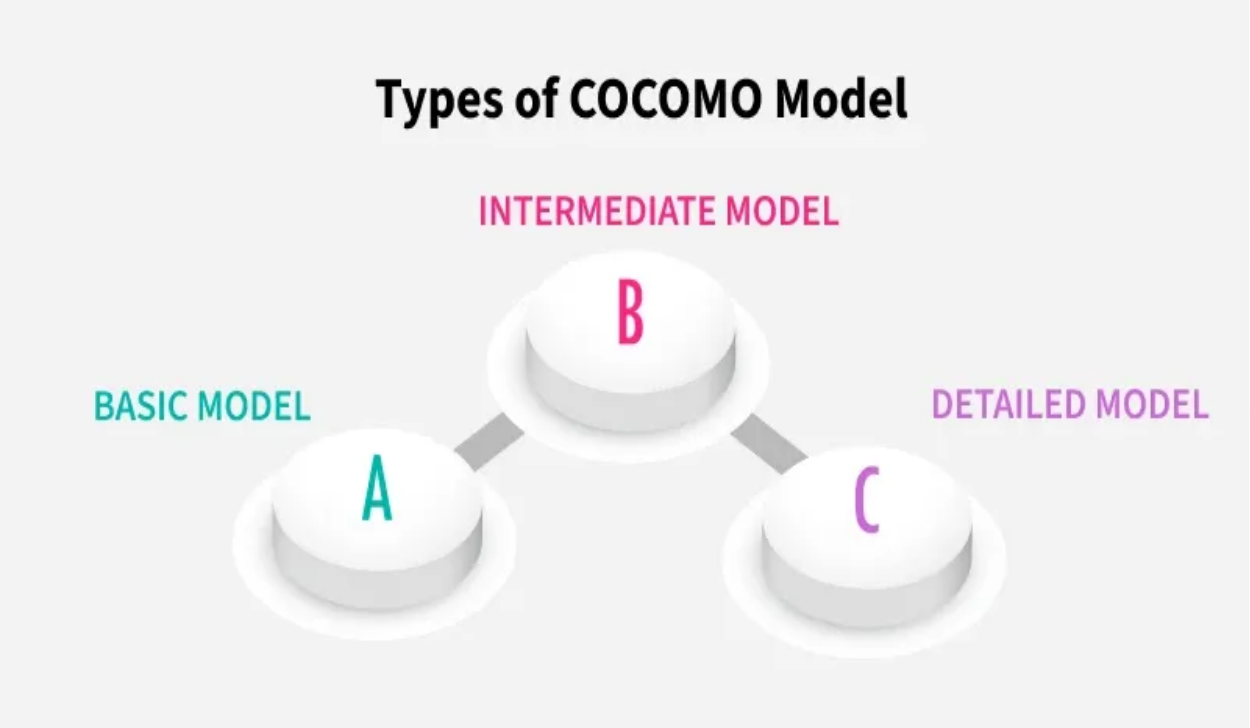
DELETE	Removes rows from a table.	Deletes rows matching a condition (WHERE). Without WHERE, all rows are deleted.	DELETE FROM users WHERE age < 18; Removes users under 18.
CREATE	Creates a new database object (e.g., table, database).	Defines the structure (e.g., table columns, data types, constraints) for the new object.	CREATE TABLE users (id INT PRIMARY KEY, name VARCHAR(50), age INT); Creates a table users with columns id, name, and age.
ALTER	Modifies an existing database object.	Changes table structure (e.g., adds/drops columns, modifies constraints).	ALTER TABLE users ADD email VARCHAR(100); Adds an email column to the users table.
DROP	Deletes a database object (e.g., table, database).	Permanently removes the specified object and its data.	DROP TABLE users; Deletes the users table and all its data.
JOIN	Combines rows from multiple tables based on a condition.	Types include INNER JOIN (matching rows), LEFT JOIN (all from left table), RIGHT JOIN (all from right table), FULL JOIN (all from both).	SELECT users.name, orders.order_id FROM users INNER JOIN orders ON users.id = orders.user_id; Retrieves user names and their order IDs.
GROUP BY	Groups rows with identical values in specified columns for aggregate calculations.	Used with aggregate functions (e.g., COUNT, SUM, AVG) to summarize data.	SELECT age, COUNT(*) FROM users GROUP BY age; Counts users by age.
HAVING	Filters grouped results.	Applies conditions to groups created by GROUP BY, similar to WHERE for rows.	SELECT age, COUNT(*) FROM users GROUP BY age HAVING COUNT(*) > 5; Shows ages with more than 5 users.
ORDER BY	Sorts query results.	Orders rows by specified columns, ascending (ASC) or descending (DESC).	SELECT name, age FROM users ORDER BY age DESC; Sorts users by age in descending order.
UNION	Combines results of two or more SELECT queries.	Merges distinct rows from multiple queries (must have same column count and types).	SELECT name FROM users UNION SELECT name FROM customers; Combines unique names from users and customers.

TRUNCATE	Removes all rows from a table without deleting the table structure.	Faster than DELETE as it resets the table without logging individual row deletions.	TRUNCATE TABLE users; Empties the users table but keeps its structure.
CREATE INDEX	Creates an index to improve query performance.	Speeds up data retrieval by creating a data structure for faster lookups.	CREATE INDEX idx_name ON users(name); Creates an index on the name column.
GRANT	Assigns permissions to users/roles.	Specifies access rights (e.g., SELECT, INSERT) for database objects.	GRANT SELECT, INSERT ON users TO 'user1'; Allows user1 to read and insert into users.
REVOKE	Removes permissions from users/roles.	Withdraws previously granted access rights.	REVOKE INSERT ON users FROM 'user1'; Removes user1's insert permission on users.

Software Engineering



Phase	Description (In Short)
Planning	Defines project scope, goals, resources, and timeline.
Defining	Specifies requirements, features, and constraints.
Designing	Creates the system architecture and detailed design.
Building	Develops the software code based on the design.
Testing	Verifies and validates the software for bugs and performance.
Deployment	Releases the software to production for use.



Need for Maintenance

Software Maintenance must be performed in order to:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.
- Requirement of user changes.
- Run the code fast

Table: Verification vs. Validation in Software Engineering

Aspect	Verification	Validation
Definition	Checks if the software meets specified requirements and design.	Confirms that the software meets the user's needs and intended purpose.
Focus	Process-oriented; ensures the product is built correctly.	Product-oriented; ensures the right product is built.
When It Occurs	Throughout the development process (e.g., design, coding).	Typically at the end, after development (e.g., testing phase).
How It Works	Involves reviews, walkthroughs, inspections, and static analysis of documents/code.	Involves dynamic testing (e.g., black box, user acceptance testing) with real-world scenarios.
Question Answered	"Are we building the product right?"	"Are we building the right product?"
Examples	- Code reviews - Unit testing - Design verification	- Alpha/Beta testing - User acceptance testing (UAT) - System testing
Techniques	Static techniques (e.g., peer reviews, checklists).	Dynamic techniques (e.g., execution, simulation).
Performed By	Developers, QA team, or design engineers.	Testers, end-users, or stakeholders.
Output	Ensures compliance with requirements and design specs.	Ensures the software satisfies user expectations.
Relation to SDLC	Part of early phases (e.g., planning, designing, building).	Part of later phases (e.g., testing, deployment).

Testing Types and their function

Testing Type	Purpose	How It Works	Notes
Alpha Test	Validates software in a controlled environment before release.	Conducted by developers or testers at the development site, simulating real-user conditions; often includes internal users.	Typically the first end-to-end test; identifies major issues.
Beta Test	Gathers feedback from real users in a live environment.	Released to a limited external audience (beta testers) after alpha; focuses on usability and bug detection in real-world use.	Often public or semi-public; prepares for full deployment.
Gamma Test	Ensures stability and performance post-beta, before final release.	Conducted after beta with a focus on refining the product based on feedback; involves final testing cycles by the development team.	Less common term; emphasizes polish and minor fixes.
Black Box Test	Tests functionality without knowledge of internal code structure.	Inputs are provided, and outputs are checked against expected results; focuses on user requirements and behavior.	Testers act as end-users; no access to code (e.g., GUI testing).
White Box Test	Tests internal code logic and structure.	Requires knowledge of the code; involves testing paths, conditions, and statements (e.g., unit testing, code coverage).	Performed by developers or specialized testers; ensures code quality.

Discrete Math

Set

- **Set**: Unordered collection of distinct objects (elements). Notation: {1, 2, 3}.
- **Element**: Object in a set. Notation: $x \in A$ (x is in A); $x \notin A$ (x not in A).
- **Empty Set**: Set with no elements. Notation: $\emptyset$ or {}.
- **Singleton Set**: Set with one element, e.g., {a}.
- **Finite Set**: Countable elements, e.g., {1, 2, 3}. Cardinality $|A| = n$.
- **Infinite Set**: Uncountable elements, e.g., natural numbers $\mathbb{N}$.
- **Subset**: $A \subseteq B$ if every element of A is in B. Proper subset $A \subset B$ if $A \subseteq B$ and $A \neq B$.
- **Universal Set**: U, all possible elements in context.
- **Power Set**: P(A), set of all subsets of A. $|P(A)| = 2^{|A|}$.
- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- **Intersection**: $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$.
- **Difference**: $A - B = \{x \mid x \in A \text{ and } x \notin B\}$.
- **Complement**: $A^c = \{x \in U \mid x \notin A\}$.
- **Cartesian Product**: $A \times B = \{(a, b) \mid a \in A, b \in B\}$.

- **De Morgan's Laws:** $(A \cup B)^c = A^c \cap B^c$; $(A \cap B)^c = A^c \cup B^c$.
- **Distributive Laws:** $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$; $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.
- **Relation:** A subset of the Cartesian product $A \times B$, defining a connection between elements of sets A and B.
- **Domain:** Set of all first elements (x) in relation $R \subseteq A \times B$.
- **Range:** Set of all second elements (y) in relation $R \subseteq A \times B$.
- **Reflexive:** For all $x \in A$, $(x, x) \in R$.
- **Symmetric:** If $(x, y) \in R$, then $(y, x) \in R$.
- **Antisymmetric:** If $(x, y) \in R$ and $(y, x) \in R$, then $x = y$.
- **Transitive:** If $(x, y) \in R$ and $(y, z) \in R$, then $(x, z) \in R$.
- **Equivalence Relation:** Relation that is reflexive, symmetric, and transitive.
- **Partial Order:** Relation that is reflexive, antisymmetric, and transitive.
- **Total Order:** Partial order where any two elements are comparable ($x \leq y$ or $y \leq x$).
- **Composition:** For relations $R \subseteq A \times B$ and $S \subseteq B \times C$, S ◆R = $\{(x, z) \mid \exists y : (x, y) \in R, (y, z) \in S\}$.
- **Inverse Relation:** $R^{-1} = \{(y, x) \mid (x, y) \in R\}$.
- **Equivalence Class:** For equivalence relation R on A, $[x] = \{y \in A \mid (x, y) \in R\}$.
- **Matrix Representation:** Relation R on $A \times A$ can be represented as a binary matrix (1 if $(x, y) \in R$, 0 otherwise).
- **Closure:** Smallest superset of R with a property (e.g., reflexive, transitive closure).

Propositional Logic and Predicate Logic

Propositional Logic

- **Proposition:** Statement that is true or false (e.g., "It is raining").
- **Connectives:**
  - **Negation (¬):** ¬p is true if p is false.
  - **Conjunction (∧):** p ∧ q is true if both p and q are true.
  - **Disjunction (∨):** p ∨ q is true if p or q (or both) is true.
  - **Implication (→):** p → q is false only if p is true and q is false.
  - **Biconditional (↔):** p ↔ q is true if p and q have the same truth value.
- **Truth Table:** Lists truth values for all possible inputs of a proposition.
- **Tautology:** Formula always true (e.g., p ∨ ¬p).
- **Contradiction:** Formula always false (e.g., p ∧ ¬p).

- **Logical Equivalence**: Two formulas with identical truth values (e.g., $\neg(p \wedge q) \equiv \neg p \vee \neg q$, De Morgan's Law).
- **Satisfiability**: A formula is satisfiable if it is true for some assignment of truth values.

Predicate Logic

- **Predicate**: Function mapping objects to true/false (e.g., P(x) = "x is even").
- **Quantifiers**:
  - **Universal (∀)**: $\forall x\ P(x)$ means P(x) is true for all x in the domain.
  - **Existential (∃)**: $\exists x\ P(x)$ means P(x) is true for at least one x in the domain.
- **Domain**: Set of all possible values for variables.
- **Free/Bound Variables**: A variable is bound if under a quantifier; otherwise, free.
- **Negation of Quantifiers**: $\neg\forall x\ P(x) \equiv \exists x\ \neg P(x)$; $\neg\exists x\ P(x) \equiv \forall x\ \neg P(x)$.
- **Nested Quantifiers**: E.g., $\forall x\ \exists y\ P(x, y)$ means for every x, there exists a y such that P(x, y) is true.
- **Validity**: Formula true in all interpretations (models).
- **Satisfiability**: Formula true in at least one interpretation.
- **Scope**: Range of a quantifier's effect in a formula.
- **Translation**: Convert natural language to logic (e.g., "All men are mortal" → $\forall x$ (Man(x) → Mortal(x))).

Functions

- **Function**: Mapping $f: A \to B$ assigning each element in A (domain) to exactly one element in B (codomain).
- **Domain**: Set of input elements A.
- **Codomain**: Set of possible output elements B.
- **Range**: Set of actual outputs $\{f(x) \mid x \in A\}$.
- **Injective (One-to-One)**: If $f(x_1) = f(x_2)$, then $x_1 = x_2$.
- **Surjective (Onto)**: Every $y \in B$ has some $x \in A$ such that $f(x) = y$.
- **Bijective**: Injective and surjective (one-to-one correspondence).
- **Composition**: $(f \circ g)(x) = f(g(x))$, for functions $g: A \to B$ and $f: B \to C$.
- **Inverse**: For bijective f, $f^{-1}: B \to A$ satisfies $f(f^{-1}(y)) = y$ and $f^{-1}(f(x)) = x$.
- **Identity Function**: $f(x) = x$ for all x in domain.
- **Floor/Ceiling**: $\lfloor x \rfloor$ = greatest integer $\leq x$; $\lceil x \rceil$ = least integer $\geq x$.

Counting Principles

· **Sum Rule**: If event A can occur in m ways and event B in n ways (disjoint), then A or B occurs in m + n ways.

· **Product Rule**: If event A can occur in m ways and event B in n ways (independent), then A and B occur in m × n ways.

· **Permutation**: Ordered arrangement of n objects taken r at a time: $P(n,r) = n! / (n-r)!$.

· **Combination**: Unordered selection of n objects taken r at a time: C(n,r) = n! / (r!(n-r)!).
· **Binomial Theorem**: $(x + y)^n = \Sigma (C(n,k) x^{(n-k)} y^k)$ from k=0 to n.
· **Inclusion-Exclusion**: For two sets: $|A \cup B| = |A| + |B| - |A \cap B|$. For three sets: $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$.
· **Pigeonhole Principle**: If n items are placed into m containers and n > m, at least one container has more than one item.
· **Generalized Pigeonhole Principle**: If n items are placed into m containers, at least one container has at least $\lceil n/m \rceil$ items.
· **Derangements**: Permutations of n objects where no object is in its original position: $!n = n! \Sigma ((-1)^k / k!)$ from k=0 to n.
· **Stars and Bars**: Number of ways to distribute n identical items into k distinct groups: C(n+k-1, k-1).

Graph Theory

- **Graph**: Collection of vertices (nodes) and edges (connections). Notation: G = (V, E).
- **Directed Graph (Digraph)**: Edges have direction (e.g., (u, v) ≠ (v, u)).
- **Undirected Graph**: Edges have no direction (e.g., {u, v} = {v, u}).
- **Simple Graph**: No loops or multiple edges between vertices.
- **Multigraph**: Allows multiple edges between vertices.
- **Degree**: Number of edges incident to a vertex. In digraphs: in-degree (incoming) and out-degree (outgoing).
- **Path**: Sequence of vertices connected by edges, no vertex repeated.
- **Cycle**: Path that starts and ends at the same vertex, no other vertices repeated.
- **Connected Graph**: Path exists between any pair of vertices.
- **Tree**: Connected graph with no cycles; has n-1 edges for n vertices.
- **Bipartite Graph**: Vertices can be divided into two sets such that edges only connect between sets.

Operating System

Category	Operating System Names
Desktop/Laptop OS	Windows (e.g., Windows 11, Windows 10, Windows 7), macOS (e.g., macOS Sonoma, Ventura, Monterey), Linux (e.g., Ubuntu, Fedora, Debian, Arch Linux, Linux Mint, Pop!_OS, Manjaro, Zorin OS), Chrome OS, Haiku
Mobile OS	Android, iOS, HarmonyOS, KaiOS, Sailfish OS, Ubuntu Touch, /e/OS, LineageOS

Server OS	Windows Server, Linux (e.g., Ubuntu Server, CentOS, Red Hat Enterprise Linux, Debian Server, SUSE Linux Enterprise Server), Unix (e.g., Solaris, AIX, HP-UX), FreeBSD, OpenBSD, NetBSD
Real-Time OS	VxWorks, QNX, FreeRTOS, RTEMS, ThreadX, Zephyr, Integrity, PikeOS
Embedded OS	Embedded Linux, Windows Embedded, Tizen, Zephyr, Yocto Project, RISC OS, Contiki, TinyOS
Distributed OS	Amoeba, Plan 9, Google's Fuchsia (experimental), Inferno, MOSIX
Legacy/Other OS	MS-DOS, BeOS, OS/2, Chrome OS, FreeBSD, OpenBSD, NetBSD, AmigaOS, Symbian, BlackBerry OS, MorphOS, ReactOS, TempleOS

Process Life Cycle

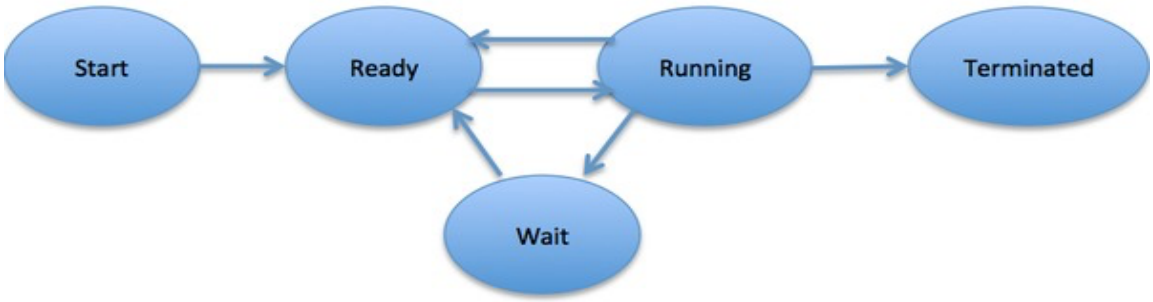
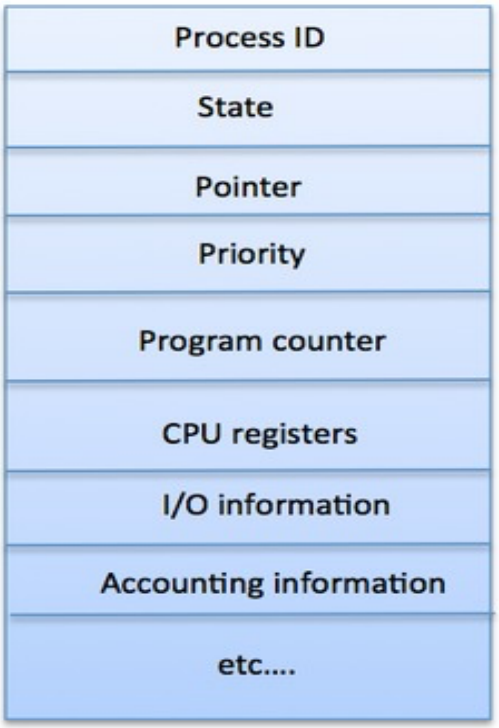


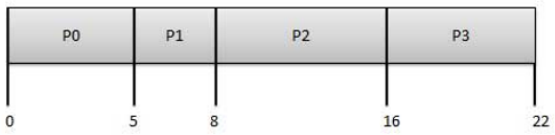
Fig: Process Control Block



S.N.	Process	Thread
1	Process is heavy weight or resource intensive.	Thread is light weight, taking lesser resources than a process.
2	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system.
3	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
5	Multiple processes without using threads use more resources.	Multiple threaded processes use fewer resources.
6	In multiple processes each process operates independently of the others.	One thread can read, write or change another thread's data.

FCFS

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



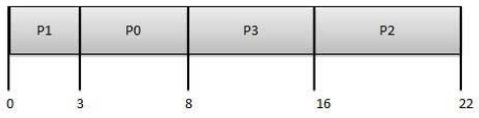
Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Process	Arrival Time	Execute Time	Service Time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



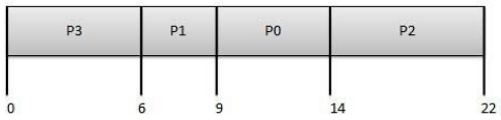
Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

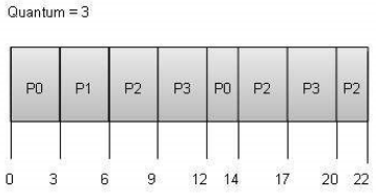
Process	Arrival Time	Execute Time	Priority	Service Time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

Round Robin Scheduling



Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Microprocessor & Interfacing

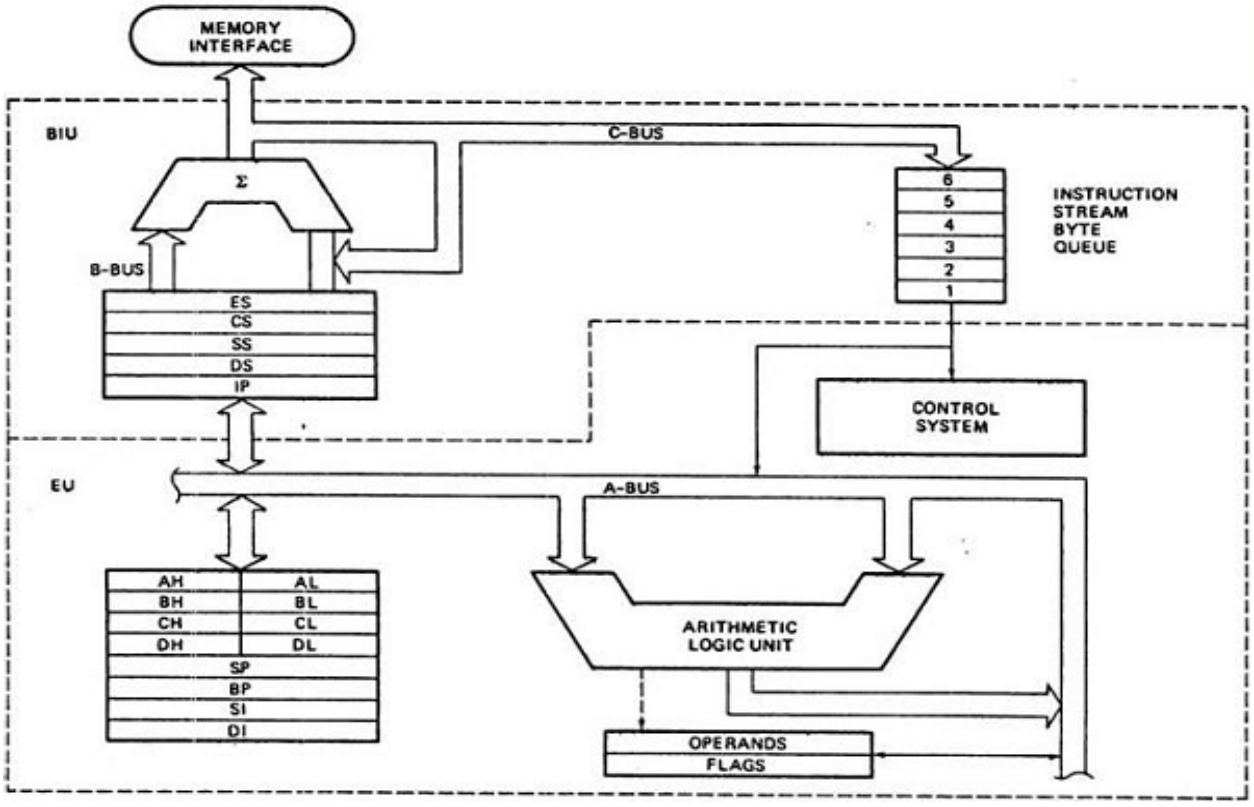


Fig: 8086 Microprocessor

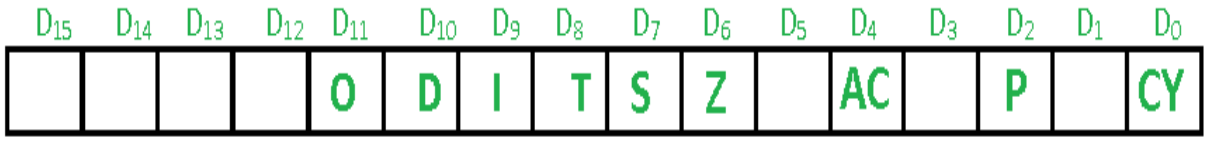


Fig: Flag Register

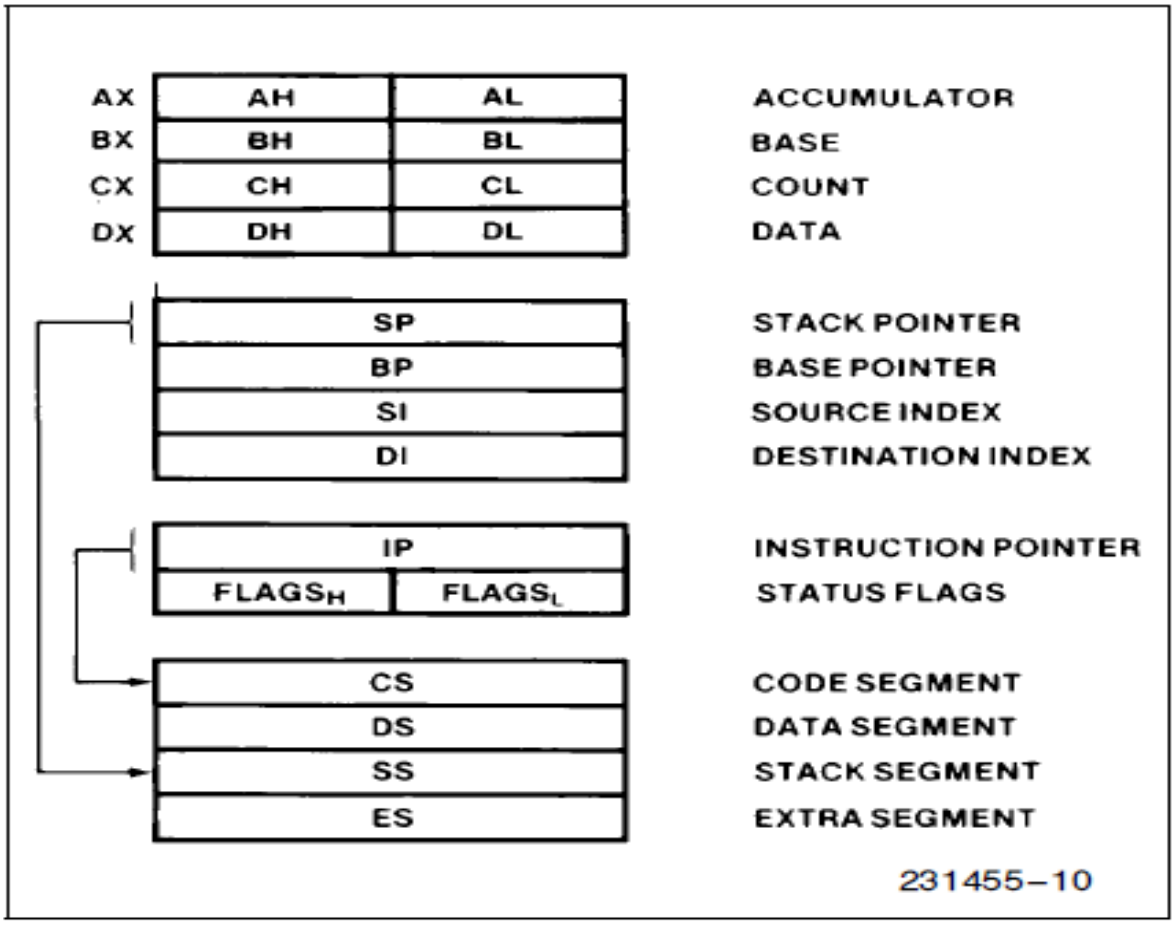
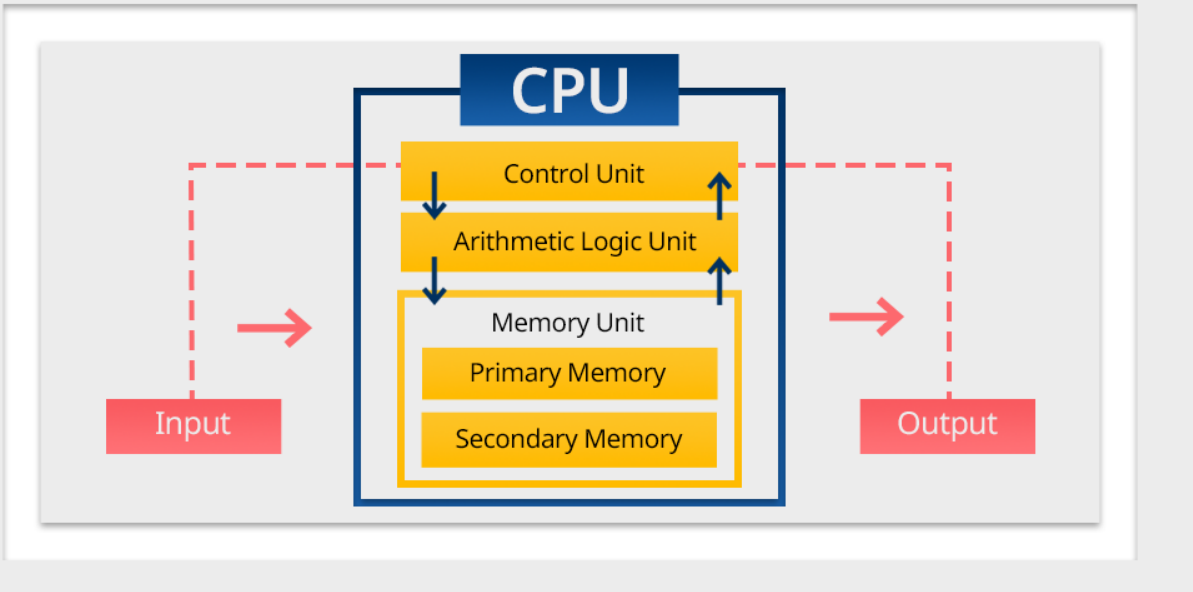
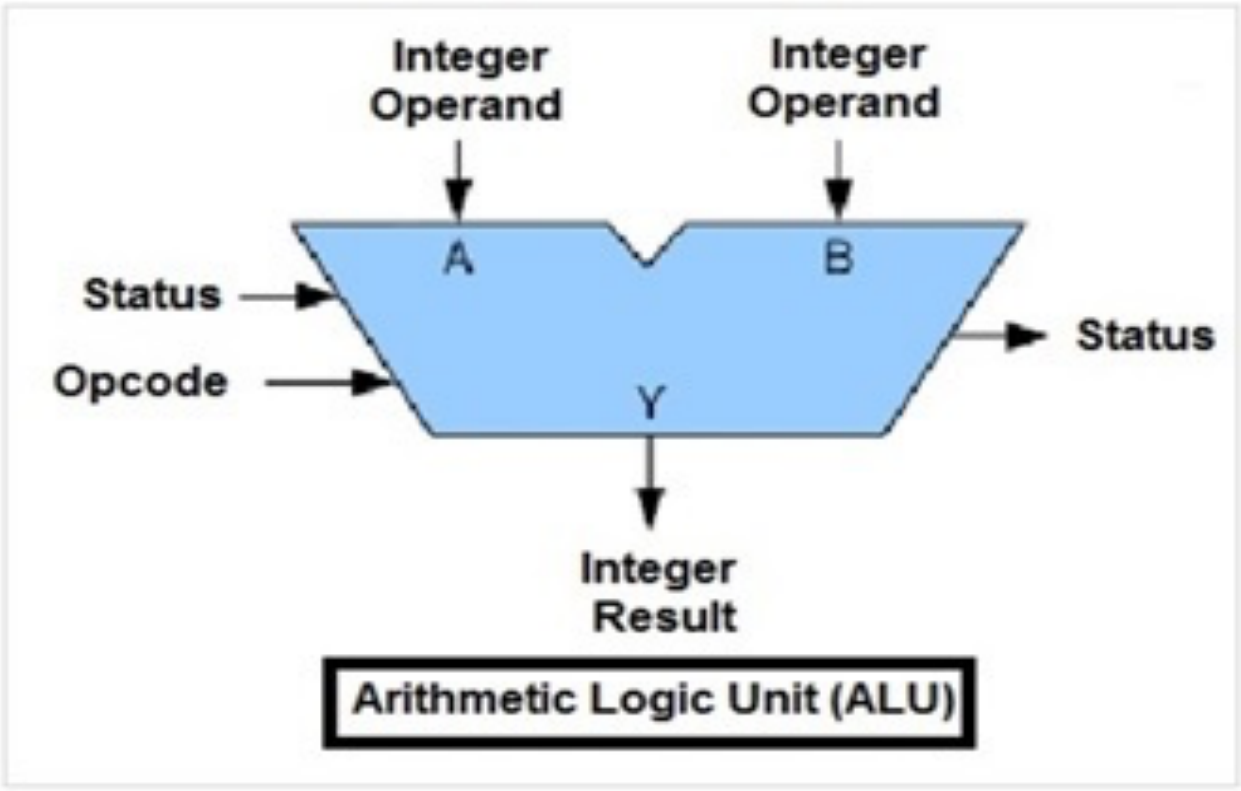


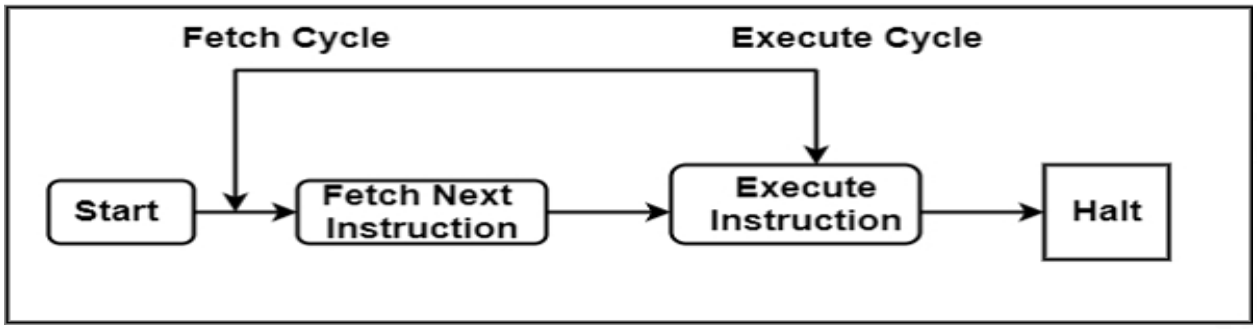
Fig: General Purpose Register

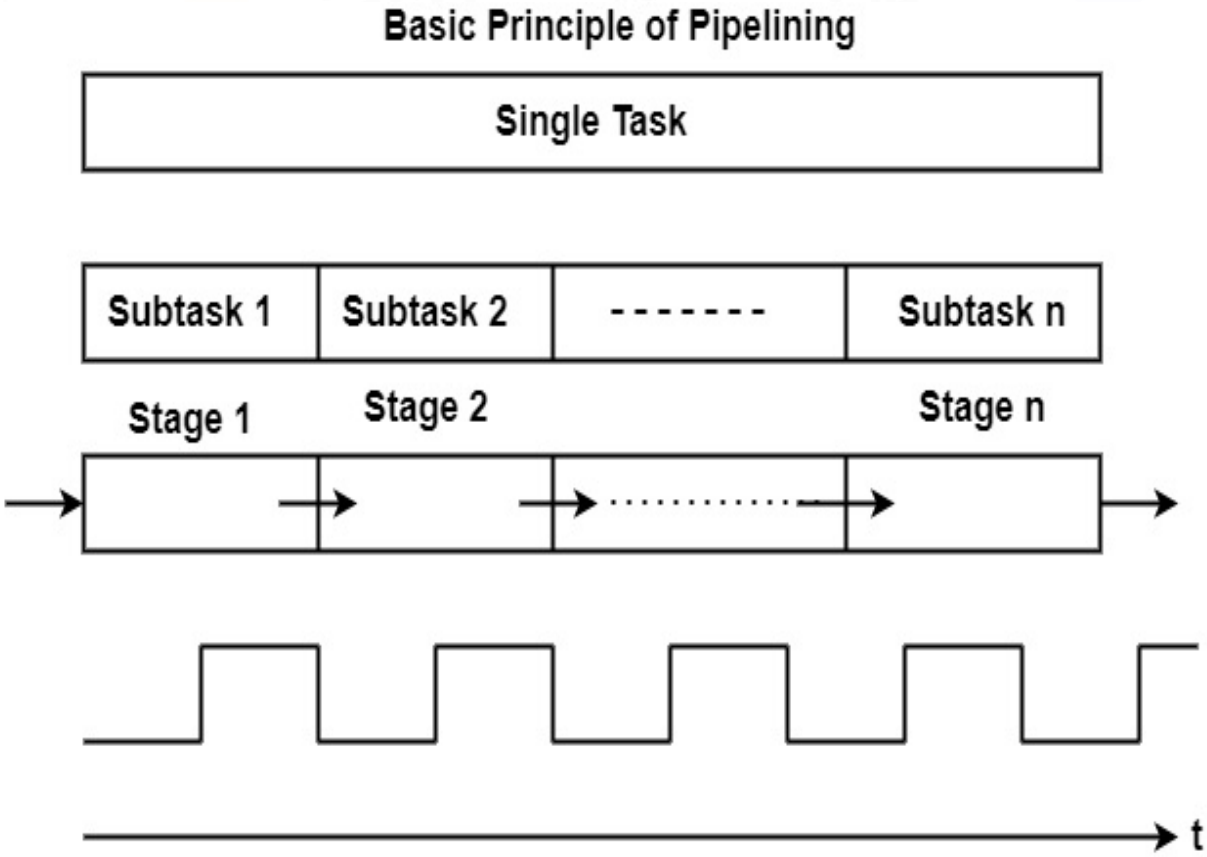
Computer Organization & Architecture





Instruction Cycle





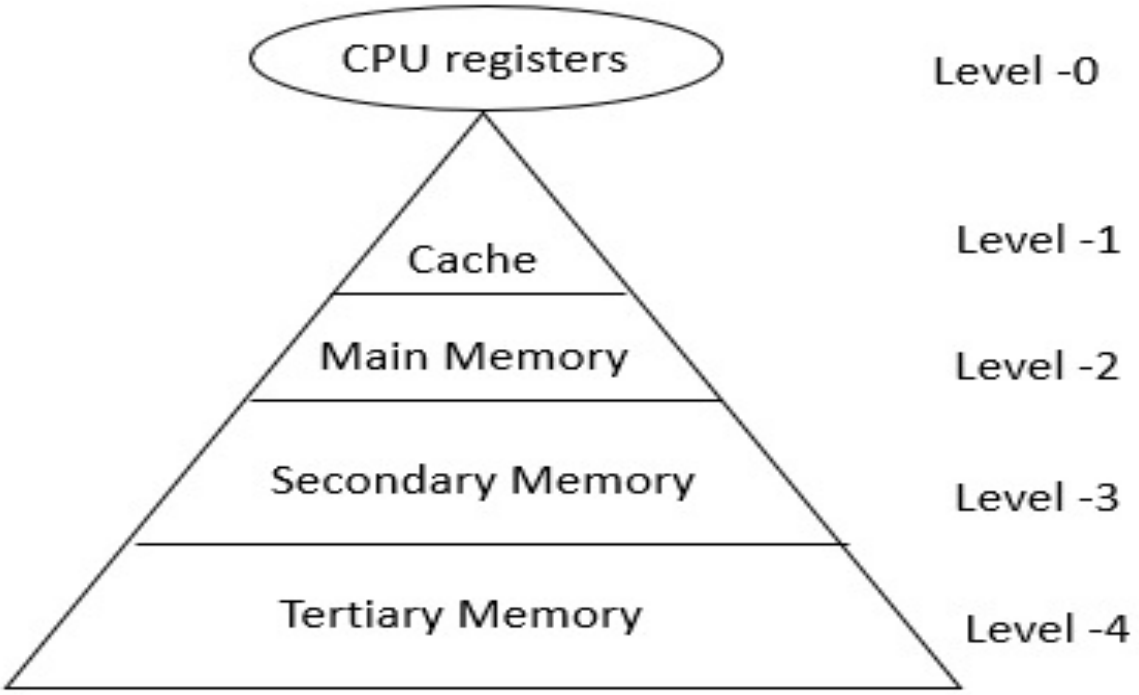


Fig: Memory Hierarchy

Level	Register	Cache	Primary memory	Secondary memory
Bandwidth	4k to 32k MB/sec	800 to 5k MB/sec	400 to 2k MB/sec	4 to 32 MB/sec
Size	Less than 1KB	Less than 4MB	Less than 2 GB	Greater than 2 GB
Access time	2 to 5nsec	3 to 10 nsec	80 to 400 nsec	5ms
Managed by	Compiler	Hardware	Operating system	OS or user

Fault-tolerance is the process of working of a system in a proper way in spite of the occurrence of the failures in the system. Even after performing the so many testing processes there is possibility of failure in system. Practically a system can't be made entirely error free. hence, systems are designed in such a way that in case of error availability and failure, system does the work properly and given correct result.

Aspect	Address Bus	Data Bus	Control Bus
Definition	Carries memory addresses from the CPU to other components.	Transfers actual data between the CPU, memory, and I/O devices.	Transmits control signals to coordinate and manage data flow.
Function	Specifies the location (address) where data is to be read or written.	Moves data (instructions or operands) to/from memory or peripherals.	Manages timing, synchronization, and operation commands (e.g., read/write).
Direction	Unidirectional (from CPU to memory/devices).	Bidirectional (data can flow both ways).	Unidirectional (from CPU to devices).
Width (Bits)	Determines maximum addressable memory (e.g., 32-bit bus = 4 GB).	Determines data transfer size (e.g., 32-bit bus = 4 bytes per cycle).	Varies; depends on control signals (e.g., 4-10 bits).
Example Usage	Sending address 0x1000 to read data from that memory location.	Transferring the value 0xABCD from memory to CPU.	Sending a "read" signal to initiate data retrieval.
Speed Impact	Limits memory capacity; wider bus increases addressable space.	Affects data throughput; wider bus allows faster transfers.	Influences timing; ensures synchronized operations.

Components Involved	CPU, Memory, Memory Controller.	CPU, Memory, I/O Devices, Cache.	CPU, Memory, I/O Devices, Clock.
Key Characteristic	Fixed size based on CPU architecture.	Can be 8, 16, 32, or 64 bits, depending on the system.	Carries signals like read/write, interrupt, or clock pulses.

Table: Address Bus, Data Bus, and Control Bus Comparison

Artifical Intelligence

What is Artificial Intelligence?

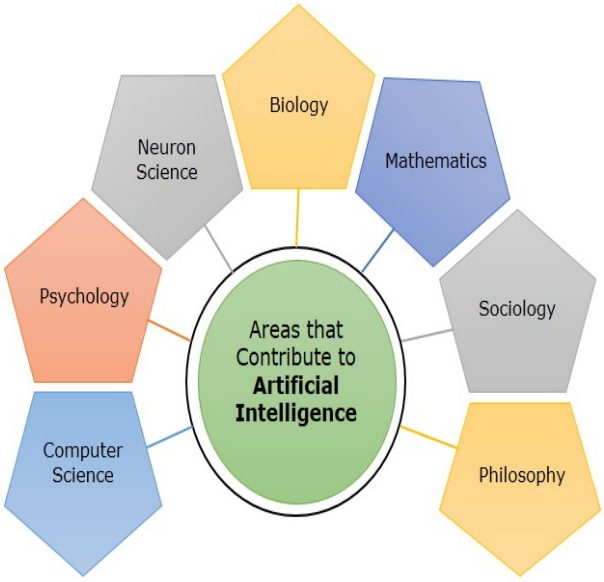
Artificial intelligence is the technology that allows systems to replicate human behavior and thoughts. At its core, AI uses algorithms to train datasets that will generate AI models that let computer systems perform tasks like recommending songs, googling route directions, or providing text translations betweentwo languages. A few examples of AI are ChatGPT, **Google Translate, Tesla, Netflix,** and many more.

According to the father of artificial intelligence, John McCarthy, it is The science and engineering of making intelligent machines, especially intelligent computer programs..

Goals of AI

The potential of AI is basically to mimic human skills and traits and apply them to machines. While the main objective of AI is to create a core technology that is able to allow computer systems to process intelligently and independently. Below are the essential goals of AI -

- To Create Expert Systems
- To Implement Human Intelligence in Machines
- To Develop Problem-Solving Ability
- To Allow Continuous Learning
- To Encourage Social Intelligence and Creativity



- **Types of Knowledge:**
  - **Explicit Knowledge:** Structured, codified data (e.g., databases, rules in PROLOG like capital(dhaka, bangladesh).).
  - **Tacit Knowledge:** Implicit, learned patterns (e.g., neural network weights for image recognition).
  - **Procedural Knowledge:** How-to knowledge (e.g., algorithms for planning or navigation).
  - **Declarative Knowledge:** Facts and statements (e.g., "Dhaka is in Bangladesh").
  - **Heuristic Knowledge:** Rule-of-thumb strategies (e.g., chess heuristics for move selection).

- **PROLOG** is a logic-based programming language where programs are expressed as facts, rules, and queries, enabling AI systems to perform automated reasoning.
  - **Purpose:** Simplifies complex tasks like knowledge base querying and symbolic inference.
  - **Relevance:** Valuable in AI for building rule-based systems, especially in domains like expert systems or semantic analysis.
- **Key Features of PROLOG:**
  - **Declarative Paradigm:** Focuses on *what* to solve (logic) rather than *how* (procedural steps).
  - **Components:**
    - **Facts:** Statements like parent(john, mary). (John is Mary's parent).
    - **Rules:** Logical implications, e.g., grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
    - **Queries:** Questions to the system, e.g., ?- grandparent(john, Z). (Who is John's grandchild?).
  - **Backtracking:** Automatically explores all possible solutions to find valid answers.
  - **Unification:** Matches variables to values (e.g., X = john binds X to john).
  - **Logic-Based:** Built on first-order predicate logic for precise reasoning.

Components	Examples
Constants	1, 5, Apple, Bob, India
Variables	x, y, z, a, b
Predicates	Father, Teacher, Brother
Functions	sqrt, LeftLegOf, AgeOf, add(x, y)
Quantifiers	∀ (for all), ∃ (there exists)
Connectives	∧ (and), ∨ (or), ¬ (not), → (implies), ↔ (if and only if)

Fig: Logical Components in Formal Logic

Expert System

The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.

Characteristics of Expert Systems

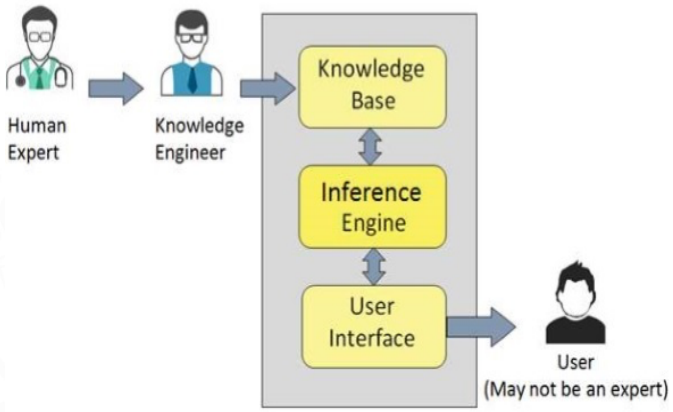
- High performance
- Understandable
- Reliable
- Highly responsive

Capabilities of Expert Systems

The expert systems are capable of -

- Advising
- Instructing and assisting human in decision making
- Demonstrating

- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem



Components of Expert Systems

The components of ES include –

- Knowledge Base
- Inference Engine
- User Interface

What is Computer Vision?

Computer Vision is a field of artificial intelligence that uses Machine Learning and Neural Networks to teach computers and systems to interpret and extract information from images and videos. This extracted information can be used for identifying and making decisions.

Hardware of Computer Vision System

Some of the most commonly used hardware components are listed below –

- Power supply
- Image acquisition device such as camera
- A processor
- A software
- A display device for monitoring the system
- Accessories such as camera stands, cables, and connectors

MCQ Part
Digital Systems

Number Systems & Codes

Q1. Decimal 173 in hexadecimal is:
a) AC b) AD
c) AF d) B3
Answer: b) AD
Explanation: Divide 173 by 16: 173 ÷ 16 = 10 remainder 13 → Hex digits A D.

Q2. Binary 110101.101_2 in decimal is:
a) 53.5 b) 53.625
c) 54.25 d) 52.625
Answer: b) 53.625
Explanation: 110101 = 53; 0.101_2 = 1/2 + 0/4 + 1/8 = 0.625 → Total = 53.625.

Q3. Excess-3 code of decimal 7 is:
a) 1010 b) 1011
c) 1000 d) 1100
Answer: a) 1010
Explanation: Excess-3 = Decimal + 3 → 7 + 3 = 10 → 1010 in binary.

Q4. Binary 1011 to Gray code is:
a) 1110 b) 1101
c) 1010 d) 1001
Answer: b) 1101
Explanation: First Gray bit = MSB, next bits = XOR(previous Gray, current binary).

Q5. BCD of decimal 39 is:
a) 100111 b) 00111001
c) 1001111 d) 01000011
Answer: b) 0011 1001
Explanation: Each decimal digit → 4-bit binary: 3 = 0011, 9 = 1001.

Boolean Algebra & Simplification

Q6. Simplify: A′B+AB+AB′
a) A + B b) A' + B
c) AB d) A + B'
Answer: a) A + B
Explanation: AB + AB' = A; then A + A'B = A + B.

Q7. Simplify: AB+A'C+BC
a) AB + C b) A + C
c) B + C d) AB + A'C
Answer: d) AB + A'C
Explanation: Apply the consensus theorem: AB + A'C + BC → AB + A'C.

Q8. Dual of A+0=Ais:
a) A · 1 = A b) A + 1 = A
c) A · 0 = A d) A' + 1 = 1
Answer: a) A · 1 = A

Q9. Complement of (AB+C) is:
a) (A' + B')C' b) A' + B' + C'
c) AB + C d) A'B' + C'
Answer: a) (A' + B')C'
Explanation: Apply De Morgan's theorem: (AB + C)' = (AB)' · C' = (A' + B')C'.

Q10. Boolean law: (A+B)(A+B')=A
a) De Morgan's law b) Absorption
c) Combining law d) Consensus
Answer: c) Combining law
Explanation: Combine terms using distributive and combining rules → A.

Logic Gates & Circuits

Q11. Y = (AB)' corresponds to which gate?
a) NAND b) NOR
c) XOR d) AND
Answer: a) NAND
Explanation: NAND = NOT(AND), exactly matches Y = (AB)'.

Q12. XOR outputs 1 when:
a) Inputs are same b) Inputs are different
c) Both inputs 0 d) Both inputs 1

Answer: b) Inputs are different
Explanation: XOR = 1 only if inputs differ.
Q13. XNOR is also called:
a) Equality gate b) Inequality gate
c) Buffer d) Inverter
Answer: a) Equality gate
Explanation: XNOR = 1 if inputs are equal.
Q14. Full adder can be implemented using:
a) 2 Half Adders + 1 OR b) 2 XOR gates
c) 2 AND gates d) 1 NAND + 1 XOR
Answer: a) 2 Half Adders + 1 OR
Explanation: Full adder = sum of two half adders + OR gate for carry.
Q15. Minimum NAND gates to implement 3-input XOR (A⊕B⊕C) is:
a) 4 b) 5
c) 6 d) 7
Answer: c) 6
Explanation: 3-input XOR = (A⊕B)⊕C; each 2-input XOR = 2 NAND + extra → total 6.

Flip-Flops & Counters

Q16. JK flip-flop race-around occurs when:
a) J=0, K=0 b) J=1, K=0
c) J=K=1, clock high d) Clock=0
Answer: c) J=K=1, clock high
Explanation: Race-around happens for level-triggered JK flip-flops when J=K=1.
Q17. D flip-flop characteristic equation:
a) Q(next) = D b) Q(next) = Q
c) Q(next) = D + Q' d) Q(next) = DQ
Answer: a) Q(next) = D
Explanation: D flip-flop transfers input D to output Q at clock edge.
Q18. Mod-10 counter requires how many flip-flops?
a) 3 b) 4
c) 5 d) 10
Answer: c) 5
Explanation: $2^4=16 > 10$ → 4 flip-flops + reset logic; minimum 5 ensures correct count.
Q19. 4-to-1 multiplexer requires how many select lines?
a) 1 b) 2
c) 3 d) 4
Answer: b) 2
Explanation: Select lines = $\log_2$(number of inputs) → $\log_2 4 = 2$.
Q20. Overflow detection in 4-bit signed addition occurs when:
a) Carry into MSB = Carry out
b) Carry into MSB ≠ Carry out
c) Sign bits equal
d) All carry zero
Answer: b) Carry into MSB ≠ Carry out
Explanation: Signed overflow = when carry into MSB differs from carry out.

Computer Programming

Introduction to Programming

Q1: Which of the following best defines a program?
a) A hardware device
b) A set of instructions given to the computer
c) An operating system
d) A compiler
Answer: b) A set of instructions given to the computer
Explanation: A program is a collection of instructions that tells the computer how to perform a task.
Q2: Which language is considered the "mother of all programming languages"?
a) Java b) Python
c) Assembly Language d) C
Answer: d) C
Explanation: C is considered the foundation

language for many modern programming languages.

Problem-Solving Techniques

Q3: What is the first step in problem-solving?
a) Writing code
b) Understanding the problem
c) Debugging
d) Testing
Answer: b) Understanding the problem
Explanation: Without fully understanding the problem, no correct solution can be designed.

Q4: Which of the following tools is commonly used to represent an algorithm visually?
a) Flowchart b) Compiler
c) Interpreter d) Debugger
Answer: a) Flowchart
Explanation: Flowcharts represent the logical sequence of steps in a diagrammatic way.

Programming Paradigms

Q5: Which paradigm emphasizes "functions" as the primary building blocks?
a) Procedural Programming
b) Object-Oriented Programming
c) Functional Programming
d) Logic Programming
Answer: c) Functional Programming
Explanation: Functional programming is based on mathematical functions and avoids state changes.

Q6: Which paradigm uses the concept of "objects and classes"?
a) Functional Programming
b) Object-Oriented Programming
c) Procedural Programming
d) Logic Programming
Answer: b) Object-Oriented Programming
Explanation: OOP revolves around objects that encapsulate both data and methods.

Structured Programming

Q7: Which of the following is NOT a feature of structured programming?
a) Sequence b) Selection
c) Inheritance d) Iteration
Answer: c) Inheritance
Explanation: Inheritance is a feature of OOP, not structured programming.

Q8: Structured programming avoids which statement for clarity?
a) while loop b) goto statement
c) if-else statement d) switch statement
Answer: b) goto statement
Explanation: Goto creates unstructured code and makes debugging difficult.

Object-Oriented Programming

Q9: Which OOP concept allows code reusability?
a) Abstraction b) Encapsulation
c) Inheritance d) Polymorphism
Answer: c) Inheritance
Explanation: Inheritance allows one class to acquire properties of another, enabling reuse.

Q10: Which OOP concept hides implementation details and shows only functionality?
a) Encapsulation b) Polymorphism
c) Abstraction d) Inheritance
Answer: c) Abstraction
Explanation: Abstraction shows only essential features while hiding unnecessary details.

Q11: Which OOP feature means "one name, multiple forms"?
a) Polymorphism b) Encapsulation
c) Inheritance d) Overloading
Answer: a) Polymorphism
Explanation: Polymorphism allows methods or operators to act differently in different contexts.

Language Basics

Q12: Which of the following is NOT a programming language?
a) Java b) HTML
c) Python d) C++
Answer: b) HTML
Explanation: HTML is a markup language, not a programming language.
Q13: Which language is platform-independent?
a) C b) Java
c) Assembly d) C++
Answer: b) Java
Explanation: Java runs on the JVM, making it platform-independent.
Q14: Which of the following is an interpreter-based language?
a) C b) C++
c) Python d) Java (compiled only)
Answer: c) Python
Explanation: Python is interpreted line by line at runtime.

Program Design

Q15: Which design approach divides a large problem into smaller sub-problems?
a) Top-down design b) Bottom-up design
c) Modular design d) Both a and c
Answer: d) Both a and c
Explanation: Top-down and modular design both aim to break problems into manageable parts.
Q16: Which of the following is NOT part of software design?
a) Algorithm design
b) Data structure design
c) Hardware fabrication
d) Modular design
Answer: c) Hardware fabrication
Explanation: Hardware fabrication belongs to hardware engineering, not programming.

Debugging

Q17: Debugging is the process of:
a) Writing code
b) Translating code into machine language
c) Finding and fixing errors in code
d) Documenting code
Answer: c) Finding and fixing errors in code
Explanation: Debugging removes logical, runtime, and syntax errors from a program.
Q18: Which type of error is most difficult to detect?
a) Syntax error b) Logical error
c) Runtime error d) Compiler error
Answer: b) Logical error
Explanation: Logical errors do not stop the program but produce incorrect results.
Q19: Which debugging tool allows step-by-step execution of a program?
a) Compiler b) IDE
c) Debugger d) Assembler
Answer: c) Debugger
Explanation: Debuggers provide step execution, breakpoints, and variable monitoring.

Q20: Which of the following is used to check correctness of an algorithm before coding?
a) Debugger b) Flowchart / Dry Run
c) Compiler d) Linker
Answer: b) Flowchart / Dry Run
Explanation: Dry run and flowcharts help validate logic before actual coding.

Q21. Which header file is required for printf() and scanf() functions?
a) stdio.h b) conio.h
c) iostream d) stdlib.h
Answer: a) stdio.h
Explanation: printf and scanf are declared in <stdio.h>.

Q22. Which storage class makes a variable visible only inside the same file?
a) auto b) static
c) extern d) register
Answer: b) static
Explanation: static at file level limits scope to the same file.

Q23. Which of the following is not a C++ OOP concept?
a) Encapsulation b) Polymorphism
c) Abstraction d) Garbage Collection
Answer: d) Garbage Collection
Explanation: C++ has no built-in garbage collection (unlike Java).

Q24. Which operator in C++ is used to allocate memory dynamically?
a) malloc b) new
c) alloc d) create
Answer: b) new
Explanation: In C++ new allocates memory, while in C we use malloc().

Q25. What is the default access specifier for members of a class in C++?
a) public b) private
c) protected d) default
Answer: b) private
Explanation: By default, class members are **private** in C++.

Q26. Which of the following is used for multiple inheritance in Java?
a) Classes b) Pointers
c) Interfaces d) Objects
Answer: c) Interfaces
Explanation: Java does not support multiple inheritance via classes, but uses **interfaces**.

Q27. Which keyword is used to prevent inheritance in Java?
a) static b) final
c) abstract d) private
Answer: b) final
Explanation: final class cannot be inherited.

Q28. Which method is the entry point of a Java program?
a) start() b) run()
c) main() d) execute()
Answer: c) main()
Explanation: JVM starts execution from public static void main(String[] args).

Q29. Which of the following languages is platform-independent?
a) C b) C++
c) Java d) Assembly
Answer: c) Java
Explanation: Java bytecode runs on JVM, making it platform-independent.

Q30. (General)
Which of the following supports both procedural and object-oriented programming?
a) C b) C++
c) Java d) Assembly
Answer: b) C++
Explanation: C++ is a **multi-paradigm** language (procedural + OOP).

Data Structures

Arrays

Q1. What is the time complexity of accessing an element in an array using its index?

a) O(1) b) O(log n)
c) O(n) d) O(n log n)

Answer: a) O(1)

Explanation: Arrays provide **direct access** using index.

Q2. Which of the following is not possible with arrays?

a) Random access
b) Sequential access
c) Dynamic size change
d) Indexing

Answer: c) Dynamic size change

Explanation: Arrays have a **fixed size** once declared.

Linked Lists

Q3. Which linked list allows traversal in both directions?

a) Singly linked list b) Doubly linked list
c) Circular linked list d) None

Answer: b) Doubly linked list

Explanation: Doubly linked lists store **previous and next pointers.**

Q4. What is the time complexity to insert a node at the beginning of a singly linked list?

a) O(1) b) O(log n)
c) O(n) d) O(n log n)

Answer: a) O(1)

Explanation: Insertion at the beginning requires **constant time.**

Stacks & Queues

Q5. Which data structure follows **LIFO**?

a) Queue b) Stack
c) Linked list d) Array

Answer: b) Stack

Explanation: Stack = Last In First Out.

Q6. Which of the following is used to implement **recursion**?

a) Queue b) Stack
c) Array d) Graph

Answer: b) Stack

Explanation: Recursive calls are stored in the **call stack.**

Q7. Which of the following is a **circular queue** problem?

a) Overflow b) Underflow
c) Both a and b d) None

Answer: c) Both a and b

Explanation: Circular queue avoids wasted space but can still have overflow/underflow issues.

Sparse / Dense Matrices

Q8. A **sparse matrix** is a matrix with:

a) Mostly non-zero elements
b) Mostly zero elements
c) Equal zero and non-zero elements
d) None

Answer: b) Mostly zero elements

Explanation: Sparse = many **zeros**, dense = many **non-zeros.**

Q9. Which representation is efficient for a sparse matrix?

a) 2D Array b) Linked List / Triplet form
c) Queue d) Stack

Answer: b) Linked List / Triplet form

Explanation: Storing only **non-zero values** saves memory.

Recursion

Q10. Which of the following is NOT an application of recursion?

a) Factorial calculation b) Binary search
c) Tower of Hanoi d) Array indexing

Answer: d) Array indexing
Explanation: Array indexing is direct, not recursive.

Q11. Recursion may cause which problem if not properly handled?
a) Stack overflow b) Queue overflow
c) Deadlock d) Memory leak
Answer: a) Stack overflow
Explanation: Too many recursive calls fill the **call stack**.

Trees

Q12. In a binary tree, maximum number of nodes at level *k* is:
a) k b) 2^k
c) 2^(k-1) d) log(k)
Answer: b) 2^k
Explanation: At level k (root = level 0), maximum = 2^k nodes.

Q13. In-order traversal of BST gives:
a) Random order b) Descending order
c) Sorted order d) Pre-order sequence
Answer: c) Sorted order
Explanation: In-order traversal of BST = ascending order.

Q14. Height of a binary tree with a single node is:
a) -1 b) 0
c) 1 d) Depends on definition
Answer: b) 0
Explanation: By definition, root alone → height = 0.

Graphs

Q15. Which representation of a graph is best for **dense graphs**?
a) Adjacency list b) Adjacency matrix
c) Linked list d) None
Answer: b) Adjacency matrix
Explanation: Dense graphs have many edges; adjacency matrix is efficient.

Q16. Which graph traversal uses **queue**?
a) DFS b) BFS
c) In-order d) Pre-order
Answer: b) BFS
Explanation: BFS = breadth-first search = uses **queue**.

Q17. Which graph traversal uses **stack (or recursion)**?
a) BFS b) DFS
c) Dijkstra's d) Kruskal's
Answer: b) DFS
Explanation: DFS is implemented using stack or recursion.

Sorting

Q18. Which sorting algorithm is based on **divide and conquer**?
a) Quick sort b) Merge sort
c) Both a and b d) Bubble sort
Answer: c) Both a and b
Explanation: Merge sort & Quick sort use **divide and conquer**.

Q19. Which sorting algorithm is **non-comparison based**?
a) Quick sort b) Heap sort
c) Radix sort d) Merge sort
Answer: c) Radix sort
Explanation: Radix sort works by **digit position**, not comparisons.

Q20. Time complexity of **heap sort** is:
a) O(n^2) b) O(n log n)
c) O(log n) d) O(n)
Answer: b) O(n log n)
Explanation: Heap sort builds heap (O(n)) + n deletions (O(log n) each).

Algorithms

Asymptotic Notations & Complexity

Q1. Which notation represents the *upper bound* of an algorithm?

a) Θ (Theta) b) Ω (Omega)
c) O (Big-O) d) o (small-o)

Answer: c) O (Big-O)

Explanation: Big-O gives **worst-case (upper bound)** performance.

Q2. Which of the following is the most efficient growth rate?

a) O(1) b) O(log n)
c) O(n) d) O(n²)

Answer: a) O(1)

Explanation: Constant time O(1) is the fastest.

Q3. If an algorithm runs in Θ(n log n), which is correct?

a) Best and worst case are linear
b) Best and worst case are quadratic
c) Both best and worst are n log n
d) Cannot determine

Answer: c) Both best and worst are n log n

Explanation: Θ notation gives **tight bound**.

Divide & Conquer

Q4. Merge sort works on which principle?

a) Greedy
b) Divide & Conquer
c) Dynamic programming
d) Backtracking

Answer: b) Divide & Conquer

Explanation: Merge sort divides array, sorts recursively, then merges.

Q5. Time complexity of Quick Sort in the **worst case** is:

a) O(n log n) b) O(n²)
c) O(log n) d) O(n)

Answer: b) O(n²)

Explanation: Worst case occurs when pivot divides array poorly.

Q6. Binary Search works only on:

a) Random data b) Unsorted data
c) Sorted data d) Linked lists

Answer: c) Sorted data

Explanation: Binary search requires **sorted array**.

Greedy Algorithms

Q7. Kruskal's algorithm is used to find:

a) Shortest path
b) Maximum flow
c) Minimum spanning tree
d) Topological order

Answer: c) Minimum spanning tree

Explanation: Kruskal builds MST by adding **lowest cost edges**.

Q8. Which greedy algorithm is used in **Huffman coding**?

a) Dynamic programming
b) Minimum spanning tree
c) Greedy merge
d) Divide & conquer

Answer: c) Greedy merge

Explanation: Huffman builds optimal codes by merging least frequent symbols.

Q9. Dijkstra's algorithm cannot handle:

a) Positive weights b) Negative weights
c) Connected graphs d) Weighted graphs

Answer: b) Negative weights

Explanation: Dijkstra fails if edge weights are **negative**.

Dynamic Programming (DP)

Q10. Which problem is best solved using DP?

a) Sorting b) Searching
c) Fibonacci sequence d) Bubble sort

Answer: c) Fibonacci sequence

Explanation: Fibonacci has **overlapping subproblems**, ideal for DP.

Q11. Bellman-Ford algorithm is based on:

a) Divide & conquer b) Greedy

c) Dynamic programming d) Backtracking

Answer: c) Dynamic programming

Explanation: Bellman-Ford relaxes edges repeatedly using DP.

Q12. Which technique is common in DP?

a) Backtracking b) Memoization

c) Randomization d) Greedy selection

Answer: b) Memoization

Explanation: Memoization stores intermediate results to avoid recomputation.

Graph Algorithms

Q13. Which algorithm is used to find shortest path in a weighted graph (with no negative weights)?

a) DFS b) BFS

c) Dijkstra d) Kruskal

Answer: c) Dijkstra

Explanation: Dijkstra finds shortest paths efficiently if weights ≥ 0.

Q14. Prim's algorithm uses which strategy?

a) Divide & conquer b) Greedy

c) Dynamic programming d) Backtracking

Answer: b) Greedy

Explanation: Prim grows MST by choosing the next minimum edge.

Q15. Floyd-Warshall algorithm finds:

a) Single source shortest path

b) All pairs shortest path

c) Minimum spanning tree

d) Maximum flow

Answer: b) All pairs shortest path

Explanation: Floyd-Warshall computes shortest paths between **all pairs** of vertices.

Q16. BFS traversal of a graph uses:

a) Stack b) Queue

c) Recursion d) Priority Queue

Answer: b) Queue

Explanation: BFS expands nodes **level by level** using a queue.

Approximation Algorithms

Q17. Approximation algorithms are mainly used for:

a) Problems solvable in polynomial time

b) NP-hard problems

c) Sorting

d) Searching

Answer: b) NP-hard problems

Explanation: Exact solutions are impractical; approximations give **near-optimal** answers.

Q18. In approximation algorithms, performance is measured by:

a) Input size b) Approximation ratio

c) Space complexity d) Stack usage

Answer: b) Approximation ratio

Explanation: It compares solution quality with the **optimal solution.**

Parallel Algorithms

Q19. Which of the following is NOT a parallel computation model?

a) PRAM b) SIMD

c) MIMD d) FIFO

Answer: d) FIFO

Explanation: FIFO is a **queue structure**, not a parallel model.

Q20. Speedup in parallel algorithms is defined as:

a) Time(sequential) ÷ Time(parallel)

b) Time(parallel) ÷ Time(sequential)

c) Parallel time × processors

d) Sequential time × processors

Answer: a) Time(sequential) ÷ Time(parallel)

Explanation: Speedup measures improvement by parallelization.

Computer Networks & Internet

OSI & TCP/IP

Q1. How many layers are there in the OSI model?

a) 5 b) 6
c) 7 d) 4

Answer: c) 7

Explanation: OSI has **7 layers**: Physical → Application.

Q2. Which layer of the OSI model handles error detection?

a) Physical b) Data Link
c) Network d) Application

Answer: b) Data Link

Explanation: Data Link ensures **error detection/correction.**

Q3. TCP/IP has how many layers?

a) 5 b) 4
c) 7 d) 6

Answer: b) 4

Explanation: TCP/IP has **4 layers**: Application, Transport, Internet, Network Access.

LAN, WAN, Ethernet

Q4. Which topology is most commonly used in Ethernet?

a) Bus b) Ring
c) Star d) Mesh

Answer: c) Star

Explanation: Modern Ethernet uses **star topology** with switches.

Q5. Which of the following is a WAN?

a) Internet b) School network
c) Home Wi-Fi d) Ethernet LAN

Answer: a) Internet

Explanation: WAN (Wide Area Network) covers **large geographical areas.**

IP Addressing, ICMP, ARP

Q6. IPv4 address size is:

a) 16 bits b) 32 bits
c) 64 bits d) 128 bits

Answer: b) 32 bits

Explanation: IPv4 uses **32-bit** addresses.

Q7. Which protocol is used for error messages in networking?

a) ARP b) ICMP
c) TCP d) UDP

Answer: b) ICMP

Explanation: ICMP reports **errors** (e.g., ping uses ICMP echo).

Q8. ARP stands for:

a) Address Resolution Protocol
b) Automatic Routing Protocol
c) Access Request Protocol
d) Application Routing Protocol

Answer: a) Address Resolution Protocol

Explanation: ARP maps **IP address → MAC address.**

Routing Protocols

Q9. RIP (Routing Information Protocol) uses which algorithm?

a) Link state b) Distance vector
c) Bellman-Ford d) Both b and c

Answer: d) Both b and c

Explanation: RIP is a **distance vector protocol** based on Bellman-Ford.

Q10. OSPF (Open Shortest Path First) is based on:

a) Distance vector b) Link state
c) Flooding d) Hierarchical routing

Answer: b) Link state

Explanation: OSPF uses **Dijkstra's shortest path** algorithm.

Network Security & Wireless

Q11. Which protocol is used for secure communication on the web?

a) HTTP b) HTTPS
c) FTP d) SMTP

Answer: b) HTTPS

Explanation: HTTPS = HTTP + **SSL/TLS encryption**.

Q12. Which IEEE standard is used for Wi-Fi?

a) 802.2 b) 802.3
c) 802.5 d) 802.11

Answer: d) 802.11

Explanation: **802.11** defines wireless LAN (Wi-Fi).

DNS, Email, VPN

Q13. DNS works on which transport protocol?

a) TCP only b) UDP only
c) Both TCP and UDP d) ICMP

Answer: c) Both TCP and UDP

Explanation: DNS uses **UDP (queries)** and **TCP (zone transfers)**.

Q14. Which protocol is used to send emails?

a) IMAP b) POP3
c) SMTP d) HTTP

Answer: c) SMTP

Explanation: SMTP is used for **sending mails**.

Q15. VPN provides:

a) Faster Internet
b) Encrypted communication
c) Extra bandwidth
d) Static IP

Answer: b) Encrypted communication

Explanation: VPN secures data via **tunneling + encryption**.

Congestion Control & ATM

Q16. Which algorithm is used in TCP congestion control?

a) Leaky bucket b) Slow start
c) Token bucket d) Round robin

Answer: b) Slow start

Explanation: TCP uses **slow start** to avoid congestion.

Q17. In congestion control, "fairness" means:

a) Each node gets equal bandwidth
b) One node gets maximum bandwidth
c) No node gets bandwidth
d) Packets are lost

Answer: a) Each node gets equal bandwidth

Explanation: Fairness ensures **balanced resource allocation**.

Q18. ATM (Asynchronous Transfer Mode) uses fixed-size cells of:

a) 32 bytes
b) 48 bytes
c) 53 bytes
d) 64 bytes

Answer: c) 53 bytes

Explanation: ATM cell = **48 bytes data + 5 bytes header** = 53 bytes.

Q19. Which layer in TCP/IP corresponds to Transport layer in OSI?

a) Application b) Internet
c) Host-to-Host d) Network access

Answer: c) Host-to-Host

Explanation: OSI **Transport layer = TCP/IP Host-to-Host**.

Q20. Which of the following is NOT a routing protocol?

a) BGP b) OSPF
c) RIP d) ARP

Answer: d) ARP

Explanation: ARP is an **address resolution protocol**, not routing.

Data Communication

Transmission Media

Q1. Which of the following has the highest bandwidth?

a) Coaxial cable b) Twisted pair
c) Optical fiber d) Radio waves

Answer: c) Optical fiber

Explanation: Optical fiber supports **very high bandwidth and speed**.

Q2. Twisted pair cables are mainly used in:

a) LAN
b) WAN
c) Satellite communication
d) Optical networks

Answer: a) LAN

Explanation: Twisted pair (Cat5/Cat6) is widely used in **Ethernet LANs**.

Signals

Q3. Digital signals are represented by:

a) Continuous values b) Discrete values
c) Sinusoidal waves d) Frequency only

Answer: b) Discrete values

Explanation: Digital signals use **0s and 1s** (discrete).

Q4. In analog signals, information is carried in:

a) Amplitude, frequency, phase
b) Only amplitude
c) Only frequency
d) Only phase

Answer: a) Amplitude, frequency, phase

Explanation: Analog signals can vary in **all three parameters**.

Modulation

Q5. ASK stands for:

a) Amplitude Shift Keying
b) Analog Signal Keying
c) Amplitude Signal Keeping
d) Analog Shift Keeping

Answer: a) Amplitude Shift Keying

Explanation: ASK modulates **amplitude** of the carrier.

Q6. PSK varies the _____ of the carrier signal.

a) Amplitude b) Phase
c) Frequency d) Power

Answer: b) Phase

Explanation: In PSK, **phase** of carrier changes.

Q7. QAM is a combination of:

a) ASK + FSK b) ASK + PSK
c) PSK + FSK d) None

Answer: b) ASK + PSK

Explanation: Quadrature Amplitude Modulation uses both **amplitude & phase**.

Q8. Manchester encoding ensures:

a) Error correction
b) Clock synchronization
c) Multiplexing
d) Modulation

Answer: b) Clock synchronization

Explanation: Manchester encoding has transitions that **help sync clock**.

Q9. In NRZ (Non-Return to Zero), binary 1 is represented by:

a) Always high b) Always low
c) Transition d) Random pulses

Answer: a) Always high

Explanation: In NRZ, **1 = high, 0 = low** (no return to zero).

Multiplexing

Q10. FDM (Frequency Division Multiplexing) divides signals based on:

a) Time slots b) Frequency bands
c) Code d) Space

Answer: b) Frequency bands

Explanation: FDM assigns **different frequency bands** to signals.

Q11. TDM (Time Division Multiplexing) works by:

a) Assigning separate frequency channels
b) Assigning separate time slots
c) Assigning separate codes
d) Assigning separate wires

Answer: b) Assigning separate time slots

Explanation: TDM gives **time slots** for each channel.

Error Detection & Flow Control

Q12. Parity check can detect:

a) All single-bit errors
b) Double-bit errors
c) Burst errors
d) No error

Answer: a) All single-bit errors

Explanation: Parity detects **odd/even single-bit errors**.

Q13. Which of the following is stronger than parity check?

a) Hamming code b) Simple checksum
c) Redundancy d) None

Answer: a) Hamming code

Explanation: Hamming code can **detect & correct errors**.

Q14. Flow control ensures:

a) Reliable addressing
b) Matching speed between sender and receiver
c) More bandwidth
d) Lower delay

Answer: b) Matching speed between sender and receiver

Explanation: Flow control prevents **buffer overflow**.

Q15. Sliding window protocol is used for:

a) Error detection b) Flow control
c) Routing d) Encoding

Answer: b) Flow control

Explanation: Sliding window ensures **efficient flow control**.

Switching Techniques

Q16. Circuit switching is mainly used in:

a) Telephone networks
b) Internet
c) Wireless LAN
d) Satellite networks

Answer: a) Telephone networks

Explanation: Circuit switching reserves a **dedicated path** for calls.

Q17. Packet switching is mainly used in:

a) Telephony b) Internet
c) Radio d) Optical signals

Answer: b) Internet

Explanation: Internet uses **packet switching (TCP/IP)**.

Q18. In packet switching, delay is caused by:

a) Dedicated circuit setup
b) Propagation only
c) Queuing and processing at routers
d) None

Answer: c) Queuing and processing at routers

Explanation: Packets may wait in queues → delay.

Mixed

Q19. Which multiplexing technique is used in telephone systems?

a) FDM b) TDM
c) CDM d) WDM

Answer: a) FDM

Explanation: Traditional telephony used **FDM channels**.

Q20. Which transmission medium is immune to EMI (Electromagnetic Interference)?

a) Coaxial b) Twisted pair
c) Optical fiber d) Wireless

Answer: c) Optical fiber

Explanation: Optical fiber uses **light**, not affected by EMI.

Database Management System

DBMS Concepts

Q1. Which of the following is NOT an advantage of DBMS?

a) Data redundancy control
b) Data integrity
c) Faster execution always
d) Data security

Answer: c) Faster execution always

Explanation: DBMS may be **slower** due to overheads, but ensures consistency & integrity.

Q2. The property of DBMS that ensures multiple users can access data without inconsistency is:

a) Security b) Concurrency control
c) Recovery d) Integrity

Answer: b) Concurrency control

Explanation: Concurrency control maintains **consistency** in multi-user systems.

ER & Relational Models

Q3. In ER modeling, a weak entity is identified by:

a) Primary key
b) Foreign key
c) Partial key + owner entity
d) Composite attribute

Answer: c) Partial key + owner entity

Explanation: Weak entities depend on **strong entities + partial key**.

Q4. A relation is in 1NF if:

a) It has a candidate key
b) It has no partial dependency
c) It has only atomic values
d) It has no transitive dependency

Answer: c) It has only atomic values

Explanation: 1NF = atomic (indivisible) values.

Normalization

Q5. 2NF removes:

a) Multi-valued attributes
b) Partial dependency
c) Transitive dependency
d) Redundancy

Answer: b) Partial dependency

Explanation: 2NF = no partial dependency.

Q6. BCNF is stronger than 3NF because:

a) Every determinant must be a candidate key
b) Every attribute is atomic
c) Every relation has only one candidate key
d) It removes multivalued dependencies

Answer: a) Every determinant must be a candidate key

Explanation: BCNF fixes anomalies not solved by 3NF.

Indexing & Query Optimization

Q7. Which indexing method is commonly used in DBMS?

a) Binary tree b) B+ tree
c) AVL tree d) Graph

Answer: b) B+ tree

Explanation: B+ tree is widely used for indexing due to balance & fast search.

Q8. Query optimization in DBMS is used to:
a) Reduce response time
b) Improve query results
c) Reduce normalization
d) Maintain integrity
Answer: a) Reduce response time
Explanation: Query optimization finds **best execution plan**.

Transaction Control

Q9. Which property of transactions ensures "all or nothing"?
a) Atomicity b) Consistency
c) Isolation d) Durability
Answer: a) Atomicity
Explanation: **Atomicity** = a transaction is executed fully or not at all.

Q10. A schedule where transactions are executed one after another without overlapping is called:
a) Serial schedule
b) Concurrent schedule
c) Strict schedule
d) Deadlock-free schedule
Answer: a) Serial schedule
Explanation: In **serial schedule**, no interleaving of operations.

Concurrency Control

Q11. Deadlock occurs when:
a) Transactions wait indefinitely
b) Transactions rollback
c) All transactions commit
d) Query optimization fails
Answer: a) Transactions wait indefinitely
Explanation: Deadlock = transactions wait for resources held by each other.

Q12. Which protocol prevents dirty read problems?
a) 2-Phase Commit b) 2-Phase Locking
c) Time Stamp Ordering d) Strict Schedule
Answer: b) 2-Phase Locking
Explanation: **2PL** ensures serializability & prevents dirty reads.

SQL Queries

Q13. Which SQL command is used to remove a table?
a) DELETE b) REMOVE
c) DROP d) ERASE
Answer: c) DROP
Explanation: DROP deletes **table structure & data**.

Q14. Which SQL clause is used to group rows with the same values?
a) ORDER BY b) GROUP BY
c) DISTINCT d) HAVING
Answer: b) GROUP BY
Explanation: GROUP BY is used with **aggregate functions**.

Q15. Which SQL statement is used to grant permissions?
a) GRANT b) GIVE
c) PERMIT d) ALLOW
Answer: a) GRANT
Explanation: GRANT provides **privileges** to users.

Implementation & Mixed

Q16. In SQL, which join returns rows when there is a match in both tables?
a) LEFT JOIN b) RIGHT JOIN
c) INNER JOIN d) FULL JOIN
Answer: c) INNER JOIN
Explanation: **INNER JOIN = only matching rows**.

Q17. Which of the following is NOT a valid SQL aggregate function?
a) SUM b) AVG
c) COUNT d) UPDATE
Answer: d) UPDATE
Explanation: UPDATE is a **DML command**, not aggregate.

Q18. A database state that satisfies all integrity constraints is called:
a) Consistent state b) Normal state
c) Atomic state d) Transaction state
Answer: a) Consistent state
Explanation: **Consistency** ensures integrity rules are satisfied.
Q19. In DBMS, view is:
a) A stored table
b) A virtual table
c) A physical copy of table
d) A file
Answer: b) A virtual table
Explanation: Views are **virtual**, created using SELECT.
Q20. Which concurrency problem occurs when two transactions read and write the same data in different order?
a) Dirty read
b) Lost update
c) Non-repeatable read
d) Phantom read
Answer: b) Lost update
Explanation: Lost update = one update overwrites another transaction's update.

Software Engineering

SDLC & Models

Q1. The first phase of the SDLC is:
a) Design
b) Requirement analysis
c) Implementation
d) Testing
Answer: b) Requirement analysis
Explanation: SDLC begins with **gathering and analyzing requirements**.
Q2. Which SDLC model is also called the "classic" model?
a) Agile b) Spiral
c) Waterfall d) V-model
Answer: c) Waterfall
Explanation: **Waterfall** is the oldest linear model.
Q3. Spiral model emphasizes:
a) Rapid delivery b) Risk analysis
c) Customer involvement d) Low cost
Answer: b) Risk analysis
Explanation: Spiral model combines iterative development with **risk management**.
Q4. Agile model focuses mainly on:
a) Documentation
b) Rigid planning
c) Customer collaboration & adaptability
d) Big design upfront
Answer: c) Customer collaboration & adaptability
Explanation: Agile is **flexible & customer-focused**.

Requirement Analysis & Design

Q5. Functional requirements define:
a) How the system works internally
b) What the system should do
c) Non-technical features
d) Maintenance plan
Answer: b) What the system should do
Explanation: **Functional = system behavior / features.**
Q6. Which diagram is NOT part of UML?
a) Class diagram b) Data Flow diagram
c) Sequence diagram d) Use case diagram
Answer: b) Data Flow diagram
Explanation: DFD is traditional, not UML.

Implementation & Testing

Q7. Unit testing focuses on:
a) Individual modules
b) Integration of modules
c) System as a whole
d) User requirements
Answer: a) Individual modules
Explanation: Unit testing checks **smallest components**.

Q8. Which testing ensures the system meets business requirements?
a) Unit testing b) Integration testing
c) Validation testing d) Regression testing
Answer: c) Validation testing
Explanation: Validation = **"Are we building the right product?"**

Q9. Regression testing is done to:
a) Test new functionality
b) Ensure changes do not break existing features
c) Optimize performance
d) Find memory leaks
Answer: b) Ensure changes do not break existing features
Explanation: Regression tests verify **old code works after updates**.

COCOMO & Estimation

Q10. COCOMO stands for:
a) Constructive Cost Model
b) Controlled Code Model
c) Corrective Cost Method
d) Computed Coding Model
Answer: a) Constructive Cost Model
Explanation: Developed by **Barry Boehm** for cost estimation.

Q11. In Basic COCOMO, software projects are classified as:
a) Organic, Semi-detached, Embedded
b) Small, Medium, Large
c) Prototype, Spiral, Agile
d) Simple, Complex, Advanced
Answer: a) Organic, Semi-detached, Embedded
Explanation: These categories depend on **complexity & team size**.

Q12. COCOMO is mainly used for estimating:
a) Testing effort
b) Maintenance cost
c) Development effort & cost
d) Software quality
Answer: c) Development effort & cost
Explanation: COCOMO predicts **effort (person-months)** & cost.

Quality Assurance & Maintenance

Q13. Software Quality Assurance (SQA) mainly ensures:
a) Faster delivery b) Lower cost
c) Software meets standards d) More features
Answer: c) Software meets standards
Explanation: SQA ensures **quality compliance**.

Q14. Which of the following is NOT a type of software maintenance?
a) Corrective b) Adaptive
c) Preventive d) Productive
Answer: d) Productive
Explanation: The main types: **Corrective, Adaptive, Perfective, Preventive**.

Q15. Corrective maintenance refers to:
a) Improving performance
b) Fixing errors
c) Adapting to environment changes
d) Preventing future issues
Answer: b) Fixing errors
Explanation: Corrective maintenance = **bug fixing**.

Q16. Preventive maintenance is done to:
a) Add new features
b) Fix existing errors
c) Reduce chances of future errors
d) Improve UI
Answer: c) Reduce chances of future errors
Explanation: Preventive = **avoiding future problems.**

Q17. Which SDLC model requires the least customer interaction?
a) Agile b) Waterfall
c) Spiral d) Incremental
Answer: b) Waterfall
Explanation: Waterfall is **rigid & sequential.**

Q18. Which testing is also called "white-box testing"?
a) Unit testing b) Integration testing
c) Structural testing d) Black-box testing
Answer: c) Structural testing
Explanation: White-box = testing **internal logic/code.**

Q19. V&V in software engineering stands for:
a) Validation & Verification
b) Version & Variation
c) Value & Volume
d) Verify & Visualize
Answer: a) Validation & Verification
Explanation: V&V = "**Are we building the product right? / Are we building the right product?**"

Q20. Which one is a key Agile principle?
a) Heavy documentation
b) Customer collaboration over contracts
c) Following strict plans
d) Big upfront design
Answer: b) Customer collaboration over contracts
Explanation: Agile values **flexibility & collaboration.**

Discrete Mathematics

Sets & Relations

Q1. If A={1,2,3},B={a,b}A = \{1,2,3\}, B = \{a,b\}A={1,2,3},B={a,b}, how many relations from A to B exist?
a) 6 b) 12
c) 64 d) 36
Answer: c) 64
Explanation: Number of relations = 2|A|×|B|=23×2=26=642^{|A|\times|B|} = 2^{3×2} = 2^6 = 642|A|×|B|=23×2=26=64.

Q2. A relation is an equivalence relation if it is:
a) Reflexive, symmetric, transitive
b) Reflexive, anti-symmetric, transitive
c) Symmetric, anti-symmetric
d) None
Answer: a) Reflexive, symmetric, transitive
Explanation: Equivalence relation = RST properties.

Logic

Q3. The contrapositive of "If it rains, then I take an umbrella" is:
a) If I take an umbrella, then it rains
b) If I don't take an umbrella, then it doesn't rain
c) If it doesn't rain, then I don't take an umbrella
d) If I don't take an umbrella, then it rains
Answer: b) If I don't take an umbrella, then it doesn't rain
Explanation: Contrapositive: "If P → Q" then "¬Q → ¬P".

Q4. Predicate logic is stronger than propositional logic because:
a) It uses AND, OR, NOT
b) It allows quantifiers (∀, ∃)

c) It is simpler
d) It has no negation
Answer: b) It allows quantifiers (∀, ∃)
Explanation: Predicates + quantifiers make it **more expressive**.

Functions & Recurrence

Q5. A function is bijective if it is:
a) One-to-one only
b) Onto only
c) Both one-to-one and onto
d) Neither
Answer: c) Both one-to-one and onto
Explanation: Bijective = **injective + surjective**.

Q6. The recurrence T(n)=2T(n/2)+nT(n) = 2T(n/2) + nT(n)=2T(n/2)+n solves to:
a) O(n) b) O(n log n)
c) O(log n) d) O(n²)
Answer: b) O(n log n)
Explanation: By Master theorem, case 2 → O(nlogn)O(n log n)O(nlogn).

Counting Principles

Q7. How many ways can 5 people be seated around a circular table?
a) 5! b) 4!
c) 3! d) 6!
Answer: b) 4!
Explanation: Circular permutation = (n-1)! = 4!.

Q8. Number of subsets of a set with 8 elements is:
a) 64 b) 128
c) 256 d) 512
Answer: c) 256
Explanation: Number of subsets = 2n=28=2562^n = 2^8 = 2562n=28=256.

Graph Theory

Q9. In a simple undirected graph with 20 vertices, the maximum number of edges is:
a) 190 b) 200
c) 180 d) 210
Answer: a) 190
Explanation: Max edges = n(n-1)/2=20×19/2=190n(n-1)/2 = 20×19/2 = 190n(n-1)/2=20×19/2=190.

Q10. Euler's formula for a connected planar graph is:
a) V − E + F = 2 b) V + E − F = 2
c) V − E − F = 2 d) V + E + F = 2
Answer: a) V − E + F = 2
Explanation: Classic **Euler's formula**.

Number Theory & Generating Functions

Q11. The GCD of 252 and 105 is:
a) 7 b) 21
c) 35 d) 63
Answer: b) 21
Explanation: By Euclidean algorithm, GCD(252, 105) = 21.

Q12. The generating function for the sequence {1,1,1,...} is:
a) 1/(1−x) b) 1/(1+x)
c) x/(1−x) d) x/(1+x)
Answer: a) 1/(1−x)
Explanation: ∑xn=1/(1-x)\sum x^n = 1/(1-x)∑xn=1/(1-x).

Numerical Analysis – Linear Systems

Q13. Gaussian elimination is used to:
a) Solve differential equations
b) Solve system of linear equations
c) Interpolate data
d) Numerical integration
Answer: b) Solve system of linear equations

Explanation: Gaussian elimination reduces system → row echelon form.

Q14. Gauss-Jordan elimination reduces matrix to:

a) Row echelon form

b) Reduced row echelon form (RREF)

c) Diagonal form only

d) Upper triangular form

Answer: b) Reduced row echelon form (RREF)

Explanation: Gauss-Jordan goes further than Gaussian → RREF.

Interpolation

Q15. Lagrange interpolation is used to:

a) Find integrals

b) Approximate a polynomial passing through given points

c) Solve recurrence

d) Find eigenvalues

Answer: b) Approximate a polynomial passing through given points

Explanation: Lagrange finds **unique polynomial** of degree (n-1).

Q16. Newton's divided difference formula is suitable for:

a) Equally spaced data points

b) Unequally spaced data points

c) Both equally and unequally

d) None

Answer: b) Unequally spaced data points

Explanation: Newton's divided difference works for **unequal intervals**.

Numerical Differentiation & Integration

Q17. Trapezoidal rule is based on approximating the curve by:

a) Straight lines b) Parabolas

c) Cubic polynomials d) Circles

Answer: a) Straight lines

Explanation: Trapezoidal → area under **straight line segments**.

Q18. Simpson's 1/3rd rule approximates the function by:

a) Straight lines b) Quadratic polynomial

c) Cubic polynomial d) Step function

Answer: b) Quadratic polynomial

Explanation: Simpson's 1/3rd rule → **parabolic arcs**.

Q19. Simpson's 3/8th rule uses:

a) 2 intervals b) 3 intervals

c) 4 intervals d) n intervals

Answer: c) 4 intervals

Explanation: Simpson's 3/8th works on **3 subintervals (4 points)**.

Q20. Which numerical integration method is generally more accurate?

a) Trapezoidal

b) Simpson's 1/3rd rule

c) Both are equal

d) None

Answer: b) Simpson's 1/3rd rule

Explanation: Simpson's 1/3rd gives **better accuracy** than trapezoidal.

Operating System

OS Overview & Types

Q1. Which of the following is NOT a type of OS?

a) Batch b) Time-sharing

c) Distributed d) Compiler

Answer: d) Compiler

Explanation: Compiler is **software**, not an OS.

Q2. Real-time OS is mainly used in:

a) Desktop computing b) Embedded systems

c) Cloud servers d) Databases

Answer: b) Embedded systems

Explanation: Real-time OS handles **time-critical tasks.**

Process & Thread Management

Q3. A process is:

a) Program in execution

b) Stored program

c) Thread of execution

d) File

Answer: a) Program in execution

Explanation: Process = **active program with resources.**

Q4. Multithreading improves:

a) CPU utilization b) Disk space

c) Memory size d) File access speed

Answer: a) CPU utilization

Explanation: Threads allow **parallel execution** → better CPU usage.

Scheduling Algorithms

Q5. Which scheduling algorithm may cause starvation?

a) FCFS

b) Round Robin

c) SJF (Shortest Job First)

d) Priority (preemptive)

Answer: d) Priority (preemptive)

Explanation: Low-priority processes may **wait indefinitely.**

Q6. In Round Robin scheduling, time slice is also called:

a) Turnaround time b) Quantum

c) Burst time d) Wait time

Answer: b) Quantum

Explanation: Quantum = **fixed CPU time per process.**

Q7. FCFS scheduling is:

a) Preemptive b) Non-preemptive

c) Priority-based d) Round robin

Answer: b) Non-preemptive

Explanation: FCFS executes processes in **arrival order** without interruption.

Inter-Process Communication & Semaphores

Q8. Which of the following is a software IPC method?

a) Shared memory b) Message passing

c) Pipes d) All of the above

Answer: d) All of the above

Explanation: IPC allows **communication between processes.**

Q9. Semaphore is used to:

a) Synchronize processes

b) Allocate memory

c) Manage files

d) Schedule jobs

Answer: a) Synchronize processes

Explanation: Semaphores prevent **race conditions.**

Q10. Deadlock occurs when:

a) Resources are overused

b) Processes wait indefinitely for each other

c) Memory is full

d) CPU utilization is low

Answer: b) Processes wait indefinitely for each other

Explanation: Deadlock = circular waiting for resources.

Q11. Banker's algorithm is used for:

a) CPU scheduling b) Memory allocation

c) Deadlock avoidance d) File access

Answer: c) Deadlock avoidance

Explanation: Banker's algorithm **prevents unsafe resource allocation.**

Memory Management

Q12. Paging eliminates:

a) Internal fragmentation

b) External fragmentation

c) CPU scheduling problems

d) File access delay
Answer: b) External fragmentation
Explanation: Paging divides memory into **fixed-size pages**.
Q13. Segmentation divides memory based on:
a) Fixed-size blocks
b) Logical segments (code, data, stack)
c) Pages
d) Files
Answer: b) Logical segments (code, data, stack)
Explanation: Segmentation = **variable-sized logical units**.
Q14. A page fault occurs when:
a) Requested page is in memory
b) Requested page is not in memory
c) Memory is full
d) Disk fails
Answer: b) Requested page is not in memory
Explanation: Page fault triggers **disk read into memory**.
Q15. Thrashing occurs due to:
a) CPU idle
b) Insufficient memory → excessive page swapping
c) Deadlock
d) High disk speed
Answer: b) Insufficient memory → excessive page swapping
Explanation: Thrashing = **system spends more time swapping than executing**.

File Management & Security

Q16. A file system maintains:
a) File name b) Metadata
c) Access permissions d) All of the above
Answer: d) All of the above
Explanation: File system tracks **data, attributes, permissions**.
Q17. RAID 1 is used for:
a) Performance only b) Fault tolerance
c) Cost saving d) Data compression
Answer: b) Fault tolerance
Explanation: RAID 1 = **mirroring for redundancy**.
Q18. Which is NOT a file access method?
a) Sequential b) Direct
c) Indexed d) Circular
Answer: d) Circular
Explanation: Sequential, direct, indexed are **standard access methods**.
Q19. In preemptive scheduling, a running process can be:
a) Interrupted by higher priority process
b) Never interrupted
c) Suspended only manually
d) Scheduled after finishing
Answer: a) Interrupted by higher priority process
Explanation: Preemptive scheduling allows **context switch**.
Q20. Mutex is different from semaphore because:
a) Mutex allows multiple processes
b) Mutex allows only **one process**
c) Mutex is for memory
d) Mutex schedules CPU
Answer: b) Mutex allows only **one process**
Explanation: Mutex = **mutual exclusion**, binary semaphore.

Microprocessor & Interfacing

8086 Architecture

Q1. 8086 is a:
a) 8-bit microprocessor
b) 16-bit microprocessor
c) 32-bit microprocessor

d) 64-bit microprocessor

Answer: b) 16-bit microprocessor

Explanation: 8086 has **16-bit data bus** and 20-bit address bus.

Q2. Maximum addressable memory of 8086 is:

a) 64 KB b) 1 MB

c) 16 MB d) 256 KB

Answer: b) 1 MB

Explanation: 8086 → 20-bit address bus → 220=1MB2^{20} = 1\text{MB}220=1MB.

Q3. Which register pair is used to generate physical address?

a) AX b) CS:IP

c) BX d) SP:BP

Answer: b) CS:IP

Explanation: Physical address = Segment × 16 + Offset.

Addressing Modes

Q4. Immediate addressing mode is:

a) Operand is in memory

b) Operand is in register

c) Operand is part of instruction

d) Operand is fetched from I/O

Answer: c) Operand is part of instruction

Explanation: Immediate = **constant inside instruction.**

Q5. Register indirect addressing uses:

a) Direct memory address

b) Register content as memory pointer

c) Immediate data

d) Segment:Offset

Answer: b) Register content as memory pointer

Explanation: Effective address = content of register.

Q6. Which addressing mode is used for stack operations?

a) Immediate b) Direct

c) Register indirect d) Indexed

Answer: c) Register indirect

Explanation: SP/BP registers hold **stack addresses.**

Instruction Set

Q7. MOV instruction is:

a) Data transfer b) Arithmetic

c) Logical d) Control

Answer: a) Data transfer

Explanation: MOV copies **data from source to destination.**

Q8. Which instruction is used for subtraction?

a) ADD b) SUB

c) INC d) DEC

Answer: b) SUB

Explanation: SUB performs **A − B** operation.

Q9. LOOP instruction is used for:

a) Arithmetic b) Logical operations

c) Repetition d) Interrupt handling

Answer: c) Repetition

Explanation: LOOP decrements CX and **jumps if CX ≠ 0.**

Memory Segmentation & TLB

Q10. 8086 divides memory into segments of size:

a) 1 KB b) 16 KB

c) 64 KB d) 256 KB

Answer: c) 64 KB

Explanation: Each segment = **64 KB.**

Q11. Physical address formula in 8086:

a) Segment + Offset

b) Segment × 16 + Offset

c) Segment × 8 + Offset

d) Segment − Offset

Answer: b) Segment × 16 + Offset

Explanation: Segment × 16 shifts **segment by 4 bits.**

Q12. TLB (Translation Lookaside Buffer) is used in:

a) Cache memory b) Virtual memory
c) Stack operations d) I/O operations

Answer: b) Virtual memory

Explanation: TLB **speeds up address translation.**

Interrupts & Stack

Q13. Software interrupt in 8086 is invoked by:

a) INT instruction b) CALL instruction
c) JMP instruction d) NOP instruction

Answer: a) INT instruction

Explanation: INT generates **software interrupt.**

Q14. Stack grows in which direction in memory?

a) Upward b) Downward
c) Left d) Right

Answer: b) Downward

Explanation: Stack pointer decreases → **stack grows downward.**

Memory-mapped I/O & Bus Interfacing

Q15. Memory-mapped I/O uses:

a) Separate address space
b) Same address space as memory
c) External bus only
d) Direct I/O instructions

Answer: b) Same address space as memory

Explanation: I/O devices are **mapped to memory addresses.**

Q16. Bus interfacing is used to connect:

a) Processor & Memory
b) Processor & I/O devices
c) Memory & I/O
d) All of the above

Answer: d) All of the above

Explanation: Bus is **shared communication path.**

Keyboard, Monitor & I/O Ports

Q17. Keyboard input in 8086 can be read using:

a) Memory-mapped I/O
b) Port-mapped I/O
c) Both a & b
d) None

Answer: c) Both a & b

Explanation: Keyboard can be interfaced via **ports or memory mapping.**

Q18. 8255 PPI is used for:

a) Interrupt handling
b) Parallel I/O interface
c) Serial communication
d) DMA

Answer: b) Parallel I/O interface

Explanation: 8255 provides **parallel input/output ports.**

Q19. Display interface in 8086 typically uses:

a) VGA controller b) LCD driver
c) Monitor port via bus d) All of the above

Answer: d) All of the above

Explanation: CPU communicates via **I/O ports & bus.**

Q20. Address lines of a microprocessor are used to:

a) Transfer data
b) Select memory or I/O locations
c) Synchronize clock
d) Handle interrupts

Answer: b) Select memory or I/O locations

Explanation: Address bus selects **memory or device locations.**

Computer Organization & Architecture

Fundamentals of Computer Design

Q1. The basic building blocks of a computer include:
a) CPU, Memory, I/O b) ALU only
c) Control unit only d) Bus only
Answer: a) CPU, Memory, I/O
Explanation: Computer = **ALU + Control Unit + Memory + I/O devices**.
Q2. The speed of a CPU is measured in:
a) Bytes b) Hertz
c) Seconds d) Bits
Answer: b) Hertz
Explanation: Clock frequency = **cycles per second (Hz)**.

ALU & Control Design

Q3. The main function of ALU is:
a) Control operations
b) Perform arithmetic & logic operations
c) Store instructions
d) Manage memory
Answer: b) Perform arithmetic & logic operations
Explanation: ALU executes **add, subtract, AND, OR, etc.**.
Q4. Control unit generates:
a) Address
b) Control signals for execution
c) Data
d) Cache
Answer: b) Control signals for execution
Explanation: Control unit **directs data flow** between CPU components.
Q5. Microprogrammed control differs from hardwired control in that it:
a) Uses ROM for control signals
b) Is faster
c) Cannot be modified
d) Uses only logic gates
Answer: a) Uses ROM for control signals
Explanation: Microprogrammed = **sequence of microinstructions stored in control memory**.

Instruction Cycle & Pipelining

Q6. The instruction cycle consists of:
a) Fetch, Decode, Execute, Store
b) Fetch only
c) Decode only
d) Execute only
Answer: a) Fetch, Decode, Execute, Store
Explanation: CPU executes instructions in **steps**.
Q7. Pipelining improves:
a) Latency b) Throughput
c) Cache size d) Memory size
Answer: b) Throughput
Explanation: Pipelining increases **instructions per unit time**.
Q8. Structural hazard occurs when:
a) Two instructions compete for the same hardware
b) Data dependency exists
c) Branch prediction fails
d) Cache misses
Answer: a) Two instructions compete for the same hardware
Explanation: Hardware conflicts cause **structural hazards**.
Q9. Data hazard arises due to:
a) Cache miss
b) Dependencies between instructions
c) Bus failure
d) Control signals
Answer: b) Dependencies between instructions

Explanation: Data hazard = **instruction depends on result of previous instruction.**

Q10. Control hazard is caused by:

a) Branch instructions

b) Data dependencies

c) Memory latency

d) ALU operations

Answer: a) Branch instructions

Explanation: Branch decisions can cause **pipeline stalls.**

Cache Memory & Memory Hierarchy

Q11. The fastest type of memory in a computer is:

a) RAM b) Cache

c) Hard disk d) Virtual memory

Answer: b) Cache

Explanation: Cache = **small, high-speed memory near CPU.**

Q12. Memory hierarchy is based on:

a) Speed, cost, size b) Color

c) Location only d) Power consumption

Answer: a) Speed, cost, size

Explanation: Faster memory = smaller, expensive; slower memory = larger, cheaper.

Q13. L1 cache is located:

a) Inside CPU b) Main memory

c) Hard disk d) External device

Answer: a) Inside CPU

Explanation: L1 = **primary cache**, very fast & small.

Systolic Arrays & Fault Tolerance

Q14. Systolic arrays are mainly used for:

a) Input/output operations

b) Parallel computation

c) Memory storage

d) Control signals

Answer: b) Parallel computation

Explanation: Systolic arrays = **data flows rhythmically through processing elements.**

Q15. Fault tolerance improves:

a) Speed b) Reliability

c) Memory size d) Cost efficiency

Answer: b) Reliability

Explanation: Redundant components or error correction **prevents system failure.**

Q16. ECC memory is used for:

a) Faster execution

b) Detecting & correcting errors

c) Large storage

d) Reducing latency

Answer: b) Detecting & correcting errors

Explanation: ECC = **Error Correction Code memory.**

Bus & Microprogrammed Control

Q17. A system bus includes:

a) Address bus, Data bus, Control bus

b) Only Data bus

c) Only Address bus

d) Only Control bus

Answer: a) Address bus, Data bus, Control bus

Explanation: Buses transfer **data, addresses, and control signals.**

Q18. Microinstructions are stored in:

a) RAM b) ROM / Control memory

c) Cache d) Registers

Answer: b) ROM / Control memory

Explanation: Control signals generated **according to microprogram.**

Q19. Hardwired control is:

a) Fixed & faster b) Flexible

c) Stored in ROM d) Slower

Answer: a) Fixed & faster

Explanation: Hardwired control uses **logic circuits**, faster but less flexible.

Q20. Pipeline stall can be reduced using:

a) Cache memory

b) Forwarding & branch prediction

c) Hard disk

d) Multithreading

Answer: b) Forwarding & branch prediction

Explanation: Techniques **avoid hazards** and improve pipeline efficiency.

Compiler & Theory of Computation

Q1. The main function of a compiler is to:

a) Execute program

b) Translate high-level code to machine code

c) Store data

d) Debug code

Answer: b) Translate high-level code to machine code

Explanation: Compiler converts **entire source code** into executable form.

Q2. Lexical analysis generates:

a) Tokens b) Parse tree

c) Symbol table d) Machine code

Answer: a) Tokens

Explanation: Lexical analysis divides source code into **identifiers, keywords, literals**.

Q3. Syntax analysis is also called:

a) Semantic analysis b) Parsing

c) Optimization d) Linking

Answer: b) Parsing

Explanation: Parsing checks **grammatical structure** of tokens.

Q4. Semantic analysis ensures:

a) Syntax correctness

b) Type compatibility & meaningful statements

c) Lexical correctness

d) Faster execution

Answer: b) Type compatibility & meaningful statements

Explanation: Example: assigning integer to string is a semantic error.

Q5. Type checking occurs at:

a) Lexical analysis b) Syntax analysis

c) Semantic analysis d) Code optimization

Answer: c) Semantic analysis

Explanation: Ensures **data types are compatible** in expressions.

Q6. Code optimization aims to:

a) Reduce code size & execution time

b) Increase syntax errors

c) Generate tokens

d) Store variables

Answer: a) Reduce code size & execution time

Explanation: Optimization improves **efficiency of generated code**.

Q7. Run-time environment handles:

a) Execution of program

b) Token generation

c) Syntax checking

d) Memory allocation for variables

Answer: d) Memory allocation for variables

Explanation: Also manages **stack, heap, and runtime errors**.

Q8. A context-free grammar is used in:

a) Lexical analysis b) Syntax analysis

c) Semantic analysis d) Code optimization

Answer: b) Syntax analysis

Explanation: CFG defines **valid statement structures**.

Q9. First step in compilation is:

a) Lexical analysis b) Parsing

c) Optimization d) Code generation

Answer: a) Lexical analysis

Explanation: Source code → **tokens** in first phase.

Q10. An intermediate code is:

a) Machine code

b) Representation between source & target code

c) Token list

d) Parse tree

Answer: b) Representation between source & target code

Explanation: Helps **code portability & optimization**.

Artificial Intelligence

Q11. AI aims to:

a) Store data

b) Simulate human intelligence

c) Execute arithmetic only

d) Manage memory

Answer: b) Simulate human intelligence

Explanation: AI enables **decision-making, learning, reasoning**.

Q12. Intelligent agents act based on:

a) Sensors & actuators b) Syntax rules

c) Lexical tokens d) Only memory

Answer: a) Sensors & actuators

Explanation: Agents **perceive environment and act** accordingly.

Q13. First-order logic allows:

a) Only propositions

b) Quantifiers and relations

c) Syntax checking

d) Only IF-ELSE rules

Answer: b) Quantifiers and relations

Explanation: FOL = **more expressive than propositional logic**.

Q14. Knowledge representation in AI is used to:

a) Store program code

b) Represent facts, rules, and relationships

c) Generate machine code

d) Optimize algorithms

Answer: b) Represent facts, rules, and relationships

Explanation: Makes AI **reasoning and decision-making possible**.

Q15. PROLOG is mainly used for:

a) Object-oriented programming

b) Logic programming & AI

c) Web development

d) Database storage

Answer: b) Logic programming & AI

Explanation: PROLOG supports **rules, facts, queries** for AI.

Q16. Expert systems are designed to:

a) Simulate human expert decisions

b) Compile code

c) Store memory

d) Manage CPU

Answer: a) Simulate human expert decisions

Explanation: Use **knowledge base + inference engine**.

Q17. NLP (Natural Language Processing) allows:

a) Computer vision

b) Understand & process human languages

c) Lexical analysis only

d) Memory optimization

Answer: b) Understand & process human languages

Explanation: Enables **chatbots, translation, speech recognition**.

Q18. Computer vision in AI is used for:

a) Arithmetic operations

b) Visual perception & image understanding

c) Lexical tokenization

d) Code optimization

Answer: b) Visual perception & image understanding

Explanation: CV allows **object recognition, image analysis**.

Q19. Inference in AI is the process of:

a) Learning data structures

b) Deduction of new facts from known facts

c) Parsing code

d) Memory allocation

Answer: b) Deduction of new facts from known facts

Explanation: Inference engine **derives conclusions** using rules.

Q20. Knowledge organization in AI helps to:

a) Optimize CPU usage

b) Efficient storage & retrieval of facts

c) Generate code

d) Perform lexical analysis

Answer: b) Efficient storage & retrieval of facts

Explanation: Knowledge structures = **semantic networks, frames, ontologies**.

The END