Updated Edition! 2026

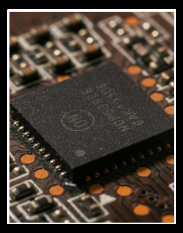
BPSC Lecturer Computer Science

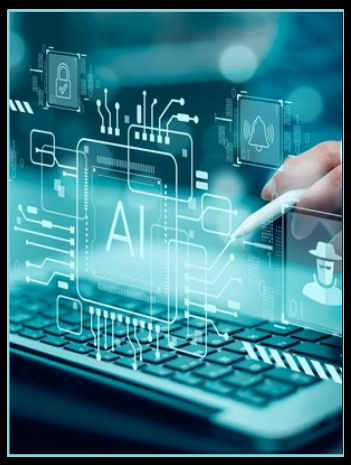
According to the Latest Syllabus & Pattern of Balochistan Public Service Commission Exams

بلوچستان کے سٹوڈنٹس کے لیے بہترین نوٹس

ALSO HELPFUL FOR FOLLOWING EXAMS:

- Subject Specialist
- Assistant Professor
- All Computer Related Jobs





Premium Features:

- To The Point, Authentic, Errorless & Result Oriented Notes + Practice MCQs
- Solved Past Papers & Most Important Questions
- According to Balochistan Public Service Commissions Syllabus
- 100% Success Guaranteed

BY
K.A. USAMA
Lecturer In Computer Science

MK PUBLICATIONS
LET'S MAKE IT HAPPEN!

For Online Order:
0348-4233593

Contact: 0342-4470091 | 0333-2605045
Email: mkpreparations@gmail.com
FOLLOW US @MK PREPARATIONS

Table of Content

Chapter 1

Information And Communication Technology (ICT) Fundamentals

Definition and Scope of ICT

Information and Communication Technology, commonly abbreviated as ICT, represents the comprehensive set of computing tools, technologies, and systems that collectively enable individuals, organizations, and societies to interact, communicate, and operate in the digital world. The term encompasses the entire spectrum of technologies used to handle telecommunications, broadcast media, intelligent building management systems, audiovisual processing and transmission systems, and network-based control and monitoring functions.

ICT is properly understood as an umbrella term that includes communication devices such as television sets and cellular phones, computer and network hardware components, satellite systems, and various associated services and applications. These applications include video conferencing platforms, distance learning systems, electronic commerce websites, and digital library services. The fundamental characteristic of ICT is its integrative nature, bringing together previously separate technologies into cohesive systems.

The concept of ICT represents the technological convergence of traditional telephone networks with modern computer networks through a single unified link system. This convergence enables the integrated transmission and processing of data, voice, and video communications across common infrastructure. For example, a single broadband connection to a home can simultaneously support telephone calls, internet browsing, and streaming video services, demonstrating the practical realization of ICT convergence.

Historical Evolution of ICT

The development of ICT as we know it today represents the culmination of several parallel technological trajectories that began to merge in the late twentieth century. The telegraph, invented in the 1830s and 1840s, represented the first electrical communication technology that separated message transmission from physical transportation. The telephone, patented by Alexander Graham Bell in 1876, added voice communication to the electrical transmission capabilities.

Radio broadcasting emerged in the early twentieth century, introducing wireless communication and mass media distribution. Television followed in the mid-twentieth century, adding visual information to broadcast capabilities. Computers, initially developed as calculating machines, evolved through several generations from vacuum tube-based systems to microprocessor-based personal computers.

The true birth of ICT occurred when digital computers gained the ability to communicate with each other through networks. The ARPANET, predecessor to the modern Internet, began operation in 1969. The development of the World Wide Web by Tim Berners-Lee in 1989-1990 provided a user-friendly interface to internet resources. The subsequent commercialization of the internet in the 1990s and the proliferation of mobile communication devices in the 2000s created the modern ICT environment.

Components of ICT

ICT comprises several interrelated components that work together to enable digital communication and information processing. The hardware component includes all physical devices such as computers, servers, routers, switches, transmission cables, satellite dishes, mobile phones, and peripheral equipment. These physical artifacts provide the tangible infrastructure upon which ICT systems are built.

Software components include operating systems, application programs, communication protocols, and middleware that enable hardware devices to function and interact. System software manages the hardware resources and provides a platform for applications. Application software performs specific user-oriented tasks such as word processing, database management, or video conferencing. Communication protocols, particularly the TCP/IP suite, define the rules and procedures for data exchange between devices.

Network infrastructure forms the connecting fabric that links devices and enables communication. This includes local area networks within buildings, wide area networks spanning geographical regions, and the global internet that interconnects networks worldwide. The infrastructure encompasses both wired technologies such as fiber optic cables and copper wires, and wireless technologies including Wi-Fi, cellular networks, and satellite communications.

Data and information constitute the content that flows through ICT systems. Raw data entered into systems becomes information through processing, organization, and contextualization. Knowledge management systems further transform information into actionable knowledge that supports decision-making and organizational learning.

Human components include users, administrators, developers, and other stakeholders who interact with ICT systems. User skills, training, and acceptance significantly influence the effectiveness of ICT implementations. Technical support personnel maintain systems and assist users. Developers create and modify software to meet evolving needs.

Applications of ICT in Modern Society

ICT has penetrated virtually every aspect of contemporary life, transforming how individuals work, learn, communicate, and entertain themselves. In education, ICT enables distance learning programs that allow students to access educational resources regardless of geographic location. Learning management systems such as Moodle and Blackboard provide platforms for course delivery, assignment submission, and student assessment. Digital libraries give students access to vast collections of academic resources. Interactive whiteboards and educational software enhance classroom instruction.

In business and commerce, ICT supports electronic commerce through online stores, payment processing systems, and digital marketing platforms. Enterprise resource planning systems integrate business processes across finance, human resources, manufacturing, and supply chain management. Customer relationship management software helps organizations track and analyze customer interactions. Video conferencing enables remote meetings and reduces travel costs and time.

Healthcare has been transformed by ICT through electronic health records that make patient information available to authorized providers regardless of location. Telemedicine allows patients in remote areas to consult with specialists. Medical imaging systems capture, store, and transmit diagnostic images. Hospital management systems coordinate patient admissions, bed assignments, and resource allocation.

Government services increasingly rely on ICT to deliver services to citizens. E-government portals allow citizens to access information, submit forms, make payments, and communicate with government agencies online. Digital identity systems provide secure authentication for government services. Open data initiatives make government information available for public use and analysis.

Banking and finance depend heavily on ICT for core operations. Automated teller machines provide 24-hour access to banking services. Electronic funds transfer systems move money between accounts instantly. Online banking allows customers to check balances, transfer funds, and pay bills from home. Stock exchanges operate entirely through computerized trading systems.

Practice MCQs

1. What is the fundamental characteristic of ICT that distinguishes it from previous technologies?
a) The use of electrical power for communication
b) The integrative nature bringing together previously separate technologies
c) The ability to store data permanently
d) The use of binary digits for data representation
Answer: The integrative nature bringing together previously separate technologies

2. According to the Von Neumann architecture, which revolutionary principle enables programs to be easily modified and treated as data?
a) The use of vacuum tubes for switching
b) Parallel processing capabilities
c) The stored-program concept where instructions and data reside in the same memory
d) The separation of control unit and arithmetic logic unit
Answer: The stored-program concept where instructions and data reside in the same memory

3. Which component of the CPU is responsible for directing and coordinating all operations by fetching, decoding, and executing instructions?
a) Arithmetic Logic Unit
b) Registers
c) Control Unit
d) Cache Memory
Answer: Control Unit

4. In the context of system characteristics, what term describes the principle that the whole system exhibits properties that cannot be understood solely by examining individual components in isolation?
a) Synergy
b) Emergence
c) Holism
d) Adaptation
Answer: Holism

5. Which type of memory is volatile, meaning it loses all stored information when power is removed, and serves as the main working memory of a computer?
a) ROM
b) Flash memory
c) RAM
d) Hard disk
Answer: RAM

6. What is the binary representation of the decimal number 43?
a) 101011
b) 110101
c) 101010
d) 111001
Answer: 101011

7. Which protocol suite provides the foundation for virtually all internet services?
a) HTTP/HTTPS
b) FTP/SMTP
c) TCP/IP
d) UDP/IP
Answer: TCP/IP

8. What is the name of the fundamental technique used for data transmission on the internet where data is broken into small pieces, each labeled with source and destination addresses?
a) Circuit switching
b) Message switching
c) Packet switching
d) Line switching
Answer: Packet switching

9. Which type of operating system is designed for time-critical applications with strict deadlines where missing a deadline may be catastrophic?

a) Multitasking operating system
b) Batch processing operating system
c) Distributed operating system
d) Real-time operating system
Answer: Real-time operating system

10. What does the CIA triad represent in cybersecurity?
a) Confidentiality, Integrity, Availability
b) Confidentiality, Intelligence, Authentication
c) Cryptography, Identity, Access
d) Control, Isolation, Authorization
Answer: Confidentiality, Integrity, Availability

11. Which cloud service model provides fundamental computing resources including virtual machines, storage, and networks where customers deploy and run arbitrary software?
a) Software as a Service
b) Platform as a Service
c) Infrastructure as a Service
d) Function as a Service
Answer: Infrastructure as a Service

12. What is the decimal equivalent of the hexadecimal number 3AF?
a) 943
b) 956
c) 927
d) 938
Answer: 943

13. Which type of network topology connects all devices to a central device such as a hub or switch, making the central device a single point of failure?
a) Bus topology
b) Ring topology
c) Mesh topology
d) Star topology
Answer: Star topology

14. What is the name of the system software that manages all hardware resources, provides services to applications, and presents a user interface?
a) Device driver
b) Utility program
c) Operating system
d) Language processor
Answer: Operating system

15. Which logic gate outputs 1 only if all its inputs are 1?
a) OR gate
b) NAND gate
c) AND gate
d) NOR gate
Answer: AND gate

16. What is the name of the process where the CPU suspends its current activities, saves its state, and transfers control to a service routine that handles signals from hardware or software?
a) Polling
b) Interrupt
c) Scheduling
d) Paging
Answer: Interrupt

17. Which encoding standard was developed to provide a universal character set capable of representing all written languages and contains over 100,000 characters?
a) ASCII
b) EBCDIC
c) Unicode
d) Extended ASCII
Answer: Unicode

18. What type of software is distributed free of charge but typically with proprietary, unmodifiable source code?
a) Open source software
b) Free software
c) Freeware
d) Shareware
Answer: Freeware

19. Which component of the computer system is fabricated on a single silicon chip

Data Communication and Computer Networks

FUNDAMENTAL CONCEPTS OF DATA COMMUNICATION

The Concept of Data Communication

In the contem+-+porary era, we live in what Marshall McLuhan famously termed the "electronic global village," where computers, telephones, wireless devices, and countless other digital instruments are interconnected across the entire planet. This unprecedented connectivity forms the backbone of modern civilization, enabling remote collaboration that transcends geographical boundaries, facilitating global e-commerce that operates twenty-four hours a day, and permitting instantaneous information sharing that has transformed how humanity accesses and disseminates knowledge. At the very heart of this interconnected world lies the fundamental discipline of **data communication**.

Data communication can be formally defined as the electronic exchange of digital information between two or more devices via some transmission medium. This definition encompasses several critical elements that merit careful examination. First, the exchange is "electronic," meaning it involves electrical signals, electromagnetic waves, or light pulses. Second, it concerns "digital information," which includes text, numbers, images, audio, video, and any other form of data that can be represented in binary form. Third, it involves "two or more devices," emphasizing the communicative nature of the process. Finally, it occurs "via a transmission medium," which may be physical (wired) or through space (wireless).

The importance of data communication in modern society cannot be overstated. Banking systems depend on secure data communication to process transactions. Healthcare institutions rely on it to share patient records and medical images. Educational institutions use it to deliver online learning. Governments employ it for administration and service delivery. In essence, every sector of contemporary society has become dependent on the reliable, efficient, and secure movement of data from one point to another.

The Five Fundamental Components of Data Communication

For any data communication system to function effectively, five essential components must be present and properly coordinated. These components constitute the foundational architecture upon which all communication systems are built, from the simplest connection between two devices to the complex infrastructure of the global Internet.

The Sender (Source or Transmitter)

The sender is the device that originates and initiates the communication process. It is responsible for generating the message and converting it into a form suitable for transmission over the chosen medium. Common examples of senders include personal computers, laptop computers, mobile telephones, tablet devices, servers, video cameras, and countless other devices capable of producing digital information. The sender performs the crucial function of transforming the internal representation of data (as stored in memory) into signals that can traverse the transmission medium. This transformation may involve encoding, modulation, or other signal processing techniques depending on the nature of the medium and the protocols being employed.

The Receiver (Destination or Sink)

The receiver is the device that accepts the transmitted message at the conclusion of the communication process. Its primary function is to capture the signal from the transmission medium and convert it back into a form that can be understood and utilized by the receiving application or user. Examples of receivers

ASCII (American Standard Code for Information Interchange)

ASCII is a 7-bit coding scheme capable of representing 128 distinct characters. It was developed in the early 1960s and became one of the most widely used character encoding standards in computing. The 128 codes include:

- Uppercase letters A through Z
- Lowercase letters a through z
- Digits 0 through 9
- Punctuation marks such as period, comma, question mark
- Control characters such as carriage return, line feed, and tab
- Special symbols such as $, %, and &

ASCII's primary limitation is that it was designed specifically for English text and cannot represent characters from other languages. Accented letters, characters from non-Latin scripts, and many special symbols simply do not exist in the ASCII set. Despite this limitation, ASCII's simplicity and efficiency have ensured its continued relevance, particularly as the foundation for more comprehensive encodings.

Unicode

Unicode is a universal character encoding standard designed to represent the writing systems of all the world's languages. Unlike ASCII's fixed 7-bit structure, Unicode uses 16 or more bits, enabling it to represent over 65,000 characters in its basic multilingual plane and over one million characters in its full extended form.

Unicode includes characters from virtually all modern and many historical scripts: Latin with its various diacritical marks, Greek, Cyrillic, Arabic, Hebrew, Devanagari, Chinese, Japanese, Korean, and hundreds more. It also includes mathematical symbols, musical notation, emoji, and countless other specialized characters.

UTF-8 is a particularly important variable-length encoding of Unicode. UTF-8 uses one to four bytes per character, with the property that the first 128 characters (the ASCII set) are encoded using a single byte identical to ASCII. This backward compatibility with ASCII has made UTF-8 the dominant encoding on the World Wide Web and in many other computing contexts. Text that consists entirely of ASCII characters is valid UTF-8, allowing systems to transition gradually from ASCII to full Unicode support.

EBCDIC (Extended Binary Coded Decimal Interchange Code)

EBCDIC is an 8-bit character encoding developed by IBM for use in its mainframe computers. Unlike ASCII, which was designed for data interchange between different manufacturers' equipment, EBCDIC was developed for IBM's specific hardware. While EBCDIC can represent 256 characters, its structure is different from ASCII, and conversion between the two encodings is sometimes required when IBM mainframes communicate with other systems. EBCDIC's use today is primarily limited to legacy IBM mainframe environments.

BCD (Binary Coded Decimal)

BCD is an older 4-bit coding scheme specifically designed to represent decimal digits (0 through 9). Each decimal digit is represented by its corresponding 4-bit binary value. For example, the decimal number 395 would be represented in BCD as 0011 1001 0101, with each group of four bits encoding a single decimal digit.

BCD was commonly used in early computing systems and in applications where exact decimal representation is important, such as financial calculations. Modern systems typically use binary

Adding a parity bit increases overhead further. This overhead makes asynchronous transmission inefficient for large data transfers.

Applications

Asynchronous transmission is simple and inexpensive to implement, making it suitable for:

- Keyboard input, where characters are generated at irregular intervals
- Low-speed data transmission
- Communication with simple devices
- Serial ports on computers (legacy RS-232 connections)

Synchronous Transmission

Synchronous transmission sends data in large, contiguous blocks, with the sender and receiver clocks synchronized for the duration of the transmission.

Mechanism

In synchronous transmission, data is organized into blocks or frames. Each frame begins with one or more synchronization characters or a special bit pattern that allows the receiver to synchronize its clock with the incoming data stream. Following the synchronization pattern, the data field contains the actual information to be transmitted, which may be hundreds or thousands of bytes. Control information, addresses, and error detection codes may also be included in the frame.

Clock Synchronization

The sender and receiver must maintain precise clock synchronization throughout the transmission of the entire frame. This synchronization may be achieved by:

- Transmitting a separate clock signal along with the data
- Encoding the clock information within the data signal itself (e.g., Manchester encoding)
- Using highly accurate clocks at both ends and synchronizing at the beginning of each frame

Efficiency

Synchronous transmission is highly efficient because overhead is limited to the synchronization pattern at the beginning of each frame, plus any control fields and error detection codes. For large data transfers, the overhead may be only a small percentage of the total bits transmitted.

Applications

Synchronous transmission is used in virtually all modern high-speed networks:

- Ethernet local area networks
- Fiber optic communication systems
- Most wide area network connections
- High-speed peripheral interfaces (USB, SATA, PCIe)

Modes of Data Transmission Based on Number of Bits

The organization of bits for transmission can be either parallel or serial, with fundamental trade-offs between speed and complexity.

Parallel Transmission

Parallel transmission sends multiple bits simultaneously over multiple parallel channels or wires.

Mechanism

In parallel transmission, a group of bits (typically a byte or word) is transmitted at the same time, with each bit using a separate wire or channel. The bits are aligned in time, so all bits of a given character are

Operation: Data from any device passes through the central device before reaching its destination. In a switched star, the central device forwards data only to the intended recipient. In a hub-based star, the central device broadcasts data to all connected devices.

Advantages:

- Easy to install and manage, with centralized control
- Failure of a single link affects only the connected device, not the entire network
- Easy to add or remove devices without disrupting the network
- Centralized troubleshooting and monitoring
- Better performance than bus topology in switched implementations

Disadvantages:

- Failure of the central device disables the entire network
- Requires more cable than bus topology (though this is less significant with modern structured cabling)
- Higher cost due to the need for central device

Example Applications: Modern Ethernet LANs almost exclusively use star topology, with switches as central devices. Home networks with a central router/switch also use star topology.

Ring Topology

Physical Layout: In a ring topology, each device is connected to exactly two other devices, forming a closed loop. Data travels around the ring from device to device, typically in one direction (unidirectional) or sometimes in both directions (bidirectional).

Operation: In a unidirectional ring, each device receives data from its upstream neighbor and forwards it to its downstream neighbor. A device copies any data addressed to it as it passes. In some ring implementations, a token passes around the ring, and only the device holding the token can transmit.

Advantages:

- Orderly network access through token passing prevents collisions
- Predictable performance under load
- Can span longer distances than bus topology
- Fair access for all devices

Disadvantages:

- Failure of a single device or cable can break the entire ring (though some implementations have redundancy)
- Adding or removing devices is disruptive, requiring the ring to be broken
- Troubleshooting is more difficult than with star topology
- Latency increases with the number of devices as data must pass through each

Example Applications: Token Ring (IEEE 802.5) and FDDI (Fiber Distributed Data Interface) are examples of ring topology implementations. Ring topology is less common today but may be found in some metropolitan area networks and legacy systems.

Mesh Topology

Physical Layout: In a mesh topology, devices are interconnected with multiple redundant paths. In a **full mesh**, every device has a dedicated point-to-point connection to every other device. In a **partial**

ICMP (Internet Control Message Protocol): ICMP is used for error reporting and diagnostic functions. It reports problems such as unreachable hosts or networks, and is used by utilities like ping and traceroute.

ARP (Address Resolution Protocol): ARP maps IP addresses to MAC addresses on the same local network. It allows a device to find the hardware address corresponding to a known IP address.

IGMP (Internet Group Management Protocol): IGMP is used for managing multicast group memberships, allowing hosts to report their multicast group memberships to nearby routers.

Layer 1: The Network Access Layer

Correspondence with OSI: The TCP/IP Network Access Layer combines the functions of OSI's Data Link and Physical layers.

Primary Function: This layer handles the physical transmission of data over the network hardware. It defines how data is formatted for transmission, how access to the physical medium is controlled, and the electrical and mechanical specifications of the interface.

Key Protocols and Technologies:

- Ethernet (IEEE 802.3) for wired LANs
- Wi-Fi (IEEE 802.11) for wireless LANs
- PPP (Point-to-Point Protocol) for direct connections
- DSL (Digital Subscriber Line) for Internet access over telephone lines
- Cable modem technologies for Internet access over cable TV infrastructure
- Fiber optic technologies such as SONET/SDH

OSI vs. TCP/IP: Critical Comparison

Feature	OSI Model	TCP/IP Model
Origin	Developed by ISO as a theoretical standard	Developed by DARPA as a practical implementation for the ARPANET
Layers	7 distinct layers	4 combined layers
Development Approach	Protocol-independent model developed first, protocols designed to fit	Protocols developed first, model derived from existing protocols
Transport Layer	Defined as connection-oriented only	Supports both connection-oriented (TCP) and connectionless (UDP)
Layer Separation	Clearly separates services, interfaces, and protocols	Does not clearly separate these concepts
Protocol Dependency	Protocol-independent, any protocol stack can be mapped to it	Tightly coupled with TCP/IP protocols
Usage	Primarily conceptual, used for teaching and design	Practical, basis of the modern Internet
Complexity	More complex with many protocols defined at each layer	Simpler, fewer protocols, more efficient
Adoption	Never fully implemented; influenced many protocol designs	Widely adopted and implemented in all Internet-connected systems

The OSI model remains valuable as a teaching tool and conceptual framework because of its clear separation of functions and its comprehensive coverage of networking concepts. The TCP/IP model,

3. **Calculate new subnet mask:** The original network mask is extended by the number of borrowed bits.
4. **Determine subnet addresses:** The subnets are formed by incrementing the borrowed bits through all possible combinations.
5. **Calculate valid host ranges:** For each subnet, the first address is the network address, the last is the broadcast address, and the addresses between are usable for hosts.

Example Problem:

Network: 192.168.1.0/24

Requirement: 5 subnets, each with at least 20 hosts

Step 1: Need ≥5 subnets → Borrow 3 bits ($2^3 = 8$ subnets)

Step 2: New prefix length = 24 + 3 = /27

Step 3: New subnet mask: 255.255.255.224 (11111111.11111111.11111111.11100000)

Step 4: Block size = 2^(32-27) = 2^5 = 32 addresses per subnet

Step 5: Subnets:

- 192.168.1.0/27 (network: 192.168.1.0, broadcast: 192.168.1.31)
- 192.168.1.32/27 (network: 192.168.1.32, broadcast: 192.168.1.63)
- 192.168.1.64/27 (network: 192.168.1.64, broadcast: 192.168.1.95)
- 192.168.1.96/27 (network: 192.168.1.96, broadcast: 192.168.1.127)
- 192.168.1.128/27 (network: 192.168.1.128, broadcast: 192.168.1.159)
- 192.168.1.160/27 (network: 192.168.1.160, broadcast: 192.168.1.191)
- 192.168.1.192/27 (network: 192.168.1.192, broadcast: 192.168.1.223)
- 192.168.1.224/27 (network: 192.168.1.224, broadcast: 192.168.1.255)

Step 6: For subnet 192.168.1.32/27:

- Network address: 192.168.1.32
- First usable host: 192.168.1.33
- Last usable host: 192.168.1.62
- Broadcast address: 192.168.1.63
- Hosts per subnet: 30 (2^5 - 2 = 30, meeting the requirement of ≥20)

IPv6: The Next Generation

IPv6 was developed to address the imminent exhaustion of IPv4 address space and to incorporate improvements based on experience with IPv4.

IPv6 Address Format

IPv6 uses 128-bit addresses, providing approximately 3.4×10^{38} addresses—enough to assign an IP address to every atom on the surface of the Earth many times over.

Addresses are written as eight groups of four hexadecimal digits, separated by colons:

Example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334

Abbreviation Rules:

- Leading zeros within a group can be omitted: 0370 → 370
- One or more consecutive groups of zeros can be replaced by "::" (but only once per address)
- 2001:0db8:85a3:0000:0000:8a2e:0370:7334 → 2001:db8:85a3::8a2e:370:7334

POP3 States:

- **Authorization:** User authentication
- **Transaction:** Client retrieves, marks for deletion
- **Update:** Server deletes marked messages

IMAP (Internet Message Access Protocol)

IMAP is used for retrieving and managing email on a server, allowing multiple clients to access the same mailbox.

Characteristics:

- Messages stored on server
- Multiple clients can access same mailbox
- Server-side folder management
- Partial fetch (can download headers first)
- TCP port 143 (unencrypted) or 993 (SSL/TLS)
- Synchronizes state across clients

IMAP Features:

- Multiple mailboxes/folders
- Message flags (read, replied, flagged)
- Searching on server
- Concurrent access with synchronization

Comparison of Email Protocols

Feature	SMTP	POP3	IMAP
Primary Use	Sending	Receiving	Receiving/Managing
Message Storage	N/A	Local (usually)	Server
Multiple Clients	N/A	Poor	Good
Server Search	N/A	No	Yes
Folder Management	N/A	Local only	Server-side
Port (plain)	25	110	143
Port (SSL)	465/587	995	993

NETWORK SECURITY FUNDAMENTALS

Security Principles (CIA Triad)

The CIA triad is the foundation of information security, representing three core objectives.

Confidentiality

Confidentiality ensures that information is accessible only to authorized parties.

Mechanisms:

- Encryption (data at rest and in transit)
- Access controls
- Authentication
- Steganography

Threats:

- Eavesdropping (packet sniffing)
- Unauthorized access

- Data leakage
- Social engineering

Integrity

Integrity ensures that information is protected from unauthorized alteration.

Mechanisms:

- Hashing (digital fingerprints)
- Message authentication codes (MAC)
- Digital signatures
- Checksums
- Version control

Threats:

- Man-in-the-middle modification
- Data corruption
- Unauthorized changes
- Replay attacks

Availability

Availability ensures that information and resources are accessible to authorized users when needed.

Mechanisms:

- Redundancy (hardware, network paths)
- Backup systems
- Load balancing
- Disaster recovery plans
- Denial-of-service protection

Threats:

- Denial-of-service (DoS) attacks
- Distributed denial-of-service (DDoS)
- Hardware failures
- Natural disasters
- Power outages

Cryptography Basics

Symmetric-Key Cryptography

Symmetric cryptography uses the same key for encryption and decryption.

Characteristics:

- Fast and efficient
- Suitable for bulk encryption
- Key must be shared securely between parties
- Key distribution problem

Algorithms:

- **AES (Advanced Encryption Standard):** Current standard, 128/192/256-bit keys
- **DES (Data Encryption Standard):** Older, 56-bit key (insecure)

Practice MCQs

1. Who famously termed the concept of the "electronic global village" where computers and wireless devices are interconnected across the planet?
a) Tim Berners-Lee
b) Vinton Cerf
c) Marshall McLuhan
d) Alan Turing
Answer: Marshall McLuhan

2. What are the five fundamental components that must be present for any data communication system to function effectively?
a) Sender, receiver, message, transmission medium, protocol
b) Transmitter, destination, data, channel, rules
c) Source, sink, information, cable, software
d) Encoder, decoder, medium, packet, address
Answer: Sender, receiver, message, transmission medium, protocol

3. Which component in a data communication system is defined as a set of pre-defined rules and conventions that govern all aspects of communication between devices?
a) Transmission medium
b) Protocol
c) Encoder
d) Interface
Answer: Protocol

4. What are the three fundamental requirements that a data communication network must satisfy to be considered functional and efficient, collectively known as the delivery requirements?
a) Speed, accuracy, reliability
b) Delivery, accuracy, timeliness
c) Bandwidth, latency, throughput
d) Security, integrity, availability
Answer: Delivery, accuracy, timeliness

5. What term refers to the variation in packet delay that can cause choppy playback in continuous media applications such as audio and video streaming?
a) Latency
b) Attenuation
c) Jitter
d) Distortion
Answer: Jitter

6. What are the three fundamental properties of an analog signal that can be varied to encode information?
a) Voltage, current, resistance
b) Amplitude, frequency, phase
c) Wavelength, velocity, period
d) Bandwidth, data rate, signal-to-noise ratio
Answer: Amplitude, frequency, phase

7. Which coding scheme is a 7-bit standard capable of representing 128 distinct characters including uppercase letters, lowercase letters, digits, and punctuation marks?
a) EBCDIC
b) Unicode
c) ASCII
d) BCD
Answer: ASCII

8. What is the name of the variable-length encoding of Unicode that uses one to four bytes per character and is backward compatible with ASCII?
a) UTF-16
b) UTF-32
c) UTF-8
d) UTF-7

Information Security

INTRODUCTION

The discipline of Information Security, often abbreviated as InfoSec, has emerged as one of the most critical domains in the contemporary computing landscape. In an era where data is regarded as the most valuable asset of organizations and individuals alike, the protection of this data from unauthorized access, use, disclosure, disruption, modification, or destruction has become paramount. The rapid evolution of technology, characterized by the proliferation of internet connectivity, cloud computing, mobile devices, and the Internet of Things, has expanded the attack surface available to malicious actors, making the study of information security not merely relevant but essential for every computer science professional.

The term Information Security refers to the practices, processes, and methodologies designed and implemented to protect information in its various forms—whether digital, physical, or intellectual—from unauthorized access and threats. It is a multidisciplinary field that draws from computer science, mathematics, management science, law, and human psychology. For the aspiring lecturer and the computer science professional, understanding information security means comprehending not only the technical mechanisms such as cryptography and access controls but also the managerial aspects involving risk assessment, policy development, and incident response.

The historical trajectory of information security is intrinsically linked to the evolution of computing itself. In the early days of mainframe computers, security was primarily physical—controlling who could access the computer room. The advent of time-sharing systems in the 1960s and 1970s introduced the need for logical access controls to separate users and protect their data from one another. The creation of the ARPANET, the predecessor to the modern Internet, by the United States Department of Defense, marked a significant turning point. As the network grew from a closed research network to a global public infrastructure, the concerns of the Department of Defense regarding confidentiality of classified information were gradually supplemented by commercial concerns regarding data integrity and, following the Morris Worm incident of 1988, the critical importance of system availability . Today, the focus has expanded to encompass the protection of personally identifiable information, driven by the explosive growth of e-commerce, social media, and stringent regulatory requirements across the globe .

These notes are designed to provide a comprehensive, examination-oriented treatment of the subject, covering all fundamental concepts, principles, and practices. The structure moves from foundational definitions through core security principles, detailed exploration of access control mechanisms, cryptographic techniques, network security measures, software and system security, to the overarching processes of risk and incident management. Each section is developed with the expectation that the reader requires no prior knowledge of the subject, building from basic definitions to advanced explanations suitable for constructing detailed subjective answers in a competitive examination setting.

FOUNDATIONS OF INFORMATION SECURITY

Defining Information Security

What is Information Security?

To understand information security, one must first understand what it means to be secure. In the context of information systems, security is not an absolute state but rather a continuous process of managing risk. When we ask the question, "When are we secure?", the most accurate answer is that we are never

External Threats originate from outside the organization. These include:

- **Hackers and Crackers**: Individuals who use their technical skills to break into systems. While "hacker" originally referred to anyone with advanced programming skills, it is now commonly used in the media to refer to malicious actors, who are more accurately termed "crackers."
- **Cybercriminals**: Organized groups that engage in cyberattacks for financial gain. They may deploy ransomware, steal credit card numbers, or engage in online fraud.
- **Nation-State Actors**: Highly sophisticated attackers sponsored by governments. Their goals may include espionage, sabotage, or influencing political events. These are often termed Advanced Persistent Threats due to their patience, resources, and persistence.
- **Hacktivists**: Individuals or groups who launch attacks to promote a political or social agenda, often defacing websites or releasing stolen documents to embarrass their targets.

Internal Threats originate from within the organization. These are often more dangerous because insiders have legitimate access to systems and a greater understanding of the organization's defenses. Internal threats can be:

- **Malicious Insiders**: Disgruntled employees or contractors who intentionally steal data, sabotage systems, or cause harm. They may act for revenge, financial gain, or to benefit a competitor.
- **Negligent Insiders**: Well-meaning employees who unintentionally cause security incidents through carelessness or lack of awareness. This includes falling for phishing emails, using weak passwords, or mishandling sensitive data.
- **Compromised Insiders**: Employees whose credentials have been stolen by external attackers. In this case, the attack originates internally, but the actor is external.

Vulnerabilities: The Weaknesses

Vulnerabilities are the openings through which threats can cause harm. They exist at multiple levels within an organization.

Technical Vulnerabilities are flaws in hardware, software, or firmware. These include:

- Software bugs and coding errors
- Unpatched operating systems and applications
- Default or weak configuration settings
- Design flaws in protocols or algorithms

Procedural Vulnerabilities are weaknesses in the processes and policies that govern how systems are used and managed. These include:

- Inadequate security policies
- Lack of security awareness training
- Poorly defined roles and responsibilities
- Failure to conduct background checks on employees

Physical Vulnerabilities are weaknesses in the physical environment. These include:

- Unlocked doors to sensitive areas
- Lack of surveillance cameras
- Poorly controlled visitor access
- Vulnerability to natural disasters due to facility location

Human Element Vulnerabilities arise from human nature itself. These include:

- Trust, which can be exploited through social engineering

Practice MCQs

1. Who famously termed the concept of the "electronic global village" where computers and wireless devices are interconnected across the planet?
a) Tim Berners-Lee
b) Vinton Cerf
c) Marshall McLuhan
d) Alan Turing
Answer: Marshall McLuhan

2. What are the five fundamental components that must be present for any data communication system to function effectively?
a) Sender, receiver, message, transmission medium, protocol
b) Transmitter, destination, data, channel, rules
c) Source, sink, information, cable, software
d) Encoder, decoder, medium, packet, address
Answer: Sender, receiver, message, transmission medium, protocol

3. Which component in a data communication system is defined as a set of pre-defined rules and conventions that govern all aspects of communication between devices?
a) Transmission medium
b) Protocol
c) Encoder
d) Interface
Answer: Protocol

4. What are the three fundamental requirements that a data communication network must satisfy to be considered functional and efficient, collectively known as the delivery requirements?
a) Speed, accuracy, reliability
b) Delivery, accuracy, timeliness
c) Bandwidth, latency, throughput
d) Security, integrity, availability
Answer: Delivery, accuracy, timeliness

5. What term refers to the variation in packet delay that can cause choppy playback in continuous media applications such as audio and video streaming?
a) Latency
b) Attenuation
c) Jitter
d) Distortion
Answer: Jitter

6. What are the three fundamental properties of an analog signal that can be varied to encode information?
a) Voltage, current, resistance
b) Amplitude, frequency, phase
c) Wavelength, velocity, period
d) Bandwidth, data rate, signal-to-noise ratio
Answer: Amplitude, frequency, phase

7. Which coding scheme is a 7-bit standard capable of representing 128 distinct characters including uppercase letters, lowercase letters, digits, and punctuation marks?
a) EBCDIC
b) Unicode
c) ASCII
d) BCD
Answer: ASCII

8. What is the name of the variable-length encoding of Unicode that uses one to four bytes per character and is backward compatible with ASCII?
a) UTF-16
b) UTF-32
c) UTF-8
d) UTF-7
Answer: UTF-8

9. Which transmission impairment refers to the loss of a signal's energy as it propagates over distance, measured in decibels?
a) Distortion
b) Noise
c) Attenuation
d) Crosstalk

of the previous unit. This differs from the decimal multiples (based on 1000) that are used in many other scientific contexts.

Kilobyte (KB) represents 1024 bytes. While some contexts, particularly hard drive manufacturers, may use kilobyte to mean exactly 1000 bytes, in computer science and programming, the binary interpretation of 1024 bytes is standard.

Megabyte (MB) consists of 1024 kilobytes, which equals 1,048,576 bytes. This unit is commonly used to measure the size of larger files such as images, audio files, and smaller software applications.

Gigabyte (GB) comprises 1024 megabytes, totaling 1,073,741,824 bytes. Modern computer RAM, storage devices, and large files such as videos are typically measured in gigabytes.

Terabyte (TB) represents 1024 gigabytes, which is approximately 1.1 trillion bytes. This unit is used for measuring the capacity of large hard drives, backup systems, and data centers.

It is important to note that these units follow a binary progression based on powers of 2, which aligns with the fundamental binary nature of computing. The International Electrotechnical Commission (IEC) has introduced alternative names such as kibibyte (KiB), mebibyte (MiB), and gibibyte (GiB) to explicitly denote binary multiples, but the traditional KB, MB, and GB notations are still widely used in programming contexts to mean binary multiples.

Binary vs. Decimal Number Systems

To fully appreciate the binary system, it is helpful to compare it systematically with the familiar decimal system. The decimal system, also called base-10, evolved naturally because humans have ten fingers and has been used for counting and calculation throughout history. The binary system, by contrast, was developed specifically for digital computing applications.

Feature	Decimal System (Base-10)	Binary System (Base-2)
Base	10	2
Digits Used	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0, 1
Place Value	Powers of 10 (..., 1000, 100, 10, 1)	Powers of 2 (..., 8, 4, 2, 1)
Smallest Unit	Digit	Bit
Example	$1980 = (1\times1000)+(9\times100)+(8\times10)+(0\times1)$	$1010_2 = (1\times8)+(0\times4)+(1\times2)+(0\times1) = 10_{10}$
Notation	Written as 45 or 45_{10}	Written with subscript, e.g., 110_2

In any binary number, the leftmost digit carries the greatest weight and is therefore called the **Most Significant Bit (MSB)** . The rightmost digit carries the least weight and is called the **Least Significant Bit (LSB)** . Understanding the position and significance of each bit is essential when performing binary arithmetic or conversions.

1.3 Number System Conversions

Converting between binary and decimal number systems is a fundamental skill in computer science. These conversions allow programmers to understand how data is stored at the machine level and to interpret binary values in human-readable form.

1.3.1 Binary to Decimal Conversion

The process of converting a binary number to its decimal equivalent uses the **positional notation method**. This method recognizes that each digit in a binary number has a positional value based on powers of 2, starting from the rightmost position which represents 2^0.

To convert a binary number to decimal, follow these steps:

1. Write the binary number with each digit clearly separated.

Floating-Point Numbers (float**)** represent numbers with decimal points. They follow the IEEE 754 standard for floating-point arithmetic, which means they have limited precision and can exhibit rounding errors.

python
price = 19.99
temperature = -5.5
scientific = 1.5e-4 *# 0.00015 in scientific notation*

Strings (str**)** represent sequences of characters enclosed in quotes. Python supports both single quotes (') and double quotes (") for string literals, which is convenient when the string itself contains quote characters.

python
name = "Ali"
message = 'Hello, World!'
multiline = """This is a
string that spans
multiple lines"""

Booleans (bool**)** represent truth values and can be either True or False. Boolean values are essential for conditional statements and logical operations.

python
is_student = True
has_graduated = False

Type Conversion

Python provides functions to convert between data types when necessary:

python
 # Convert string to integer
num_str = "123"
num_int = int(num_str)

 # Convert integer to float
num_float = float(456)

 # Convert number to string
num = 789
num_str = str(num)

 # Convert to boolean
bool(0) *# False*
bool(1) *# True*
bool("") *# False*
bool("abc") *# True*

Type Checking

The type() function returns the type of any object:

The floor division operator // performs division and rounds down to the nearest integer. The modulus operator % returns the remainder after division. These are particularly useful for problems involving cycles, grouping, or divisibility.

Comparison Operators

Comparison operators compare two values and return a Boolean result (True or False):

Operator	Description	Example	Result
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	5 > 3	True
<	Less than	5 < 3	False
>=	Greater than or equal	5 >= 5	True
<=	Less than or equal	5 <= 3	False

Logical Operators

Logical operators combine Boolean expressions:

Operator	Description	Example	Result
and	True if both are true	(5 > 3) and (10 > 5)	True
or	True if at least one is true	(5 > 3) or (10 < 5)	True
not	Inverts the Boolean value	not (5 > 3)	False

Membership Operators

Membership operators test whether a value is present in a sequence:

Operator	Description	Example	Result
in	True if value is in sequence	"a" in "apple"	True
not in	True if value is not in sequence	"x" not in "apple"	True

Control Structures

Conditional Statements

Python uses if, elif, and else for conditional execution. The syntax requires colons after each condition and indentation to indicate which statements belong to each block:

python

```
number = int(input("Enter a number: "))

if number > 0:
    print("Positive number")
elif number < 0:
    print("Negative number")
else:
    print("Zero")
```

Multiple conditions can be combined using logical operators:

python

```
age = int(input("Enter age: "))
if age >= 18 and age <= 65:
    print("Working age adult")
elif age < 18:
    print("Minor")
```

```
    </tr>
    <tr>
      <td>Alia</td>
      <td>9</td>
      <td>A</td>
    </tr>
    <tr>
      <td>Bilal</td>
      <td>10</td>
      <td>B+</td>
    </tr>
</table>
```

- <table>: The container for the entire table
- <tr>: Table row
- <th>: Table header cell (bold and centered by default)
- <td>: Table data cell

Table Sections

Tables can be divided into three sections for better structure and styling:

```
html
<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Department</th>
      <th>Salary</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Ahmed</td>
      <td>Engineering</td>
      <td>75,000</td>
    </tr>
    <tr>
      <td>Fatima</td>
      <td>Marketing</td>
      <td>65,000</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="2">Total</td>
```

Loops

for **loop:** When the number of iterations is known:

javascript

```
for (let i = 0; i < 5; i++) {
   console.log(i);  // 0, 1, 2, 3, 4
}
```

while **loop:** Repeats while a condition is true:

javascript

```
let count = 0;
while (count < 5) {
   console.log(count);
   count++;
}
```

do-while **loop:** Always executes at least once:

javascript

```
let i = 0;
do {
   console.log(i);
   i++;
} while (i < 5);
```

Loop control:

javascript

```
// break - exit loop completely
for (let i = 0; i < 10; i++) {
   if (i === 5) break;
   console.log(i);  // 0, 1, 2, 3, 4
}

// continue - skip to next iteration
for (let i = 0; i < 5; i++) {
   if (i === 2) continue;
   console.log(i);  // 0, 1, 3, 4
}
```

Arrays

Arrays in JavaScript are ordered collections that can hold values of any type.

Creating Arrays

javascript

```
// Array literal
let fruits = ["Apple", "Banana", "Orange"];

// Using Array constructor
let numbers = new Array(1, 2, 3, 4, 5);
```

```
    # Handle specific exception type
    print("That's not a valid number!")
except ZeroDivisionError:
    print("Cannot divide by zero!")
except Exception as e:
    # Handle any other exception
    print(f"An error occurred: {e}")
else:
    # Runs if no exception occurred
    print("Operation completed successfully")
finally:
    # Always runs
    print("Execution finished")
```

SUMMARY OF KEY PROGRAMMING CONSTRUCTS

The following table compares fundamental programming constructs across JavaScript and Python:

Concept	JavaScript Example	Python Example
Variable	let count = 5;	count = 5
Output	console.log(count);	print(count)
Input	let name = prompt("Name?");	name = input("Name? ")
Conditional	if (count > 10) { ... } else { ... }	if count > 10: ... else: ...
for Loop	for(let i=0; i<5; i++) { ... }	for i in range(5): ...
Array/List	let arr = [1, 2, 3];	lst = [1, 2, 3]
Function	function sum(a,b) { return a+b; }	def sum(a, b): return a + b

Practice MCQs

1. What is the fundamental reason computers use the binary number system rather than the decimal system?
a) Binary numbers are easier for humans to understand
b) The decimal system was not invented when computers were developed
c) Electronic circuits can only reliably distinguish between two states (ON and OFF)
d) Binary numbers take less space in memory
Answer: Electronic circuits can only reliably distinguish between two states (ON and OFF)

2. What is the term for the smallest unit of data in computing, which can hold only one of two possible values?
a) Byte
b) Nibble
c) Bit
d) Word
Answer: Bit

3. How many different combinations can a single byte represent?
a) 128
b) 256
c) 512
d) 1024
Answer: 256

4. What is the decimal equivalent of the binary number 101101?
a) 43
b) 45
c) 47
d) 49
Answer: 45

5. Convert the decimal number 172 to binary.
a) 10101100
b) 11001100
c) 10101010
d) 11110000
Answer: 10101100

6. In a binary number, what is the leftmost digit called?
a) Least Significant Bit (LSB)
b) Most Significant Bit (MSB)
c) High Order Bit
d) Primary Bit
Answer: Most Significant Bit (MSB)

7. Which character encoding scheme uses 7 bits to represent 128 distinct characters including uppercase and lowercase letters, digits, and punctuation marks?
a) Unicode
b) EBCDIC
c) ASCII
d) UTF-8
Answer: ASCII

8. What is the decimal value for the lowercase letter 'a' in ASCII?
a) 65
b) 97
c) 49
d) 32
Answer: 97

9. Which encoding system is backward compatible with ASCII while using variable-length encoding of one to four bytes per character?
a) UTF-16
b) UTF-32
c) UTF-8
d) EBCDIC
Answer: UTF-8

10. In the RGB color model, how many bits are typically used for each color component to achieve "true color" with approximately 16.7 million colors?
a) 4 bits
b) 6 bits
c) 8 bits
d) 16 bits
Answer: 8 bits

11. According to the Nyquist-Shannon

Object Oriented Programming in C++

INTRODUCTION TO OBJECT ORIENTED PROGRAMMING

The Evolution of Programming Paradigms

Programming paradigms have evolved significantly over the decades to address the growing complexity of software systems. Understanding this evolution is fundamental to appreciating why Object Oriented Programming emerged as a dominant paradigm.

Procedural Programming (Early Days): In the earliest days of computing, programs were written as simple sequences of instructions. As problems grew more complex, the procedural paradigm emerged, organizing code into functions or procedures that operate on data. Languages like C, FORTRAN, and Pascal exemplify this approach. The primary characteristic is the separation of data and functions—data structures exist independently, and functions manipulate them.

Structured Programming: This refinement of procedural programming introduced better control flow through constructs like loops and conditionals, replacing arbitrary jumps (goto statements). It emphasized modularity through functions but maintained the fundamental separation of data and behavior.

Limitations of Procedural Approach: As software systems grew larger, several problems became apparent:

- Data and functions remained separate, making it difficult to model real-world entities
- Global data could be accessed and modified by any function, leading to unintended side effects
- Code reuse was limited and often resulted in duplication
- Maintenance became increasingly difficult as programs grew in size
- There was no direct way to model relationships between real-world entities

The Object Oriented Solution: Object Oriented Programming emerged to address these limitations by unifying data and the operations that manipulate that data into single entities called objects. This paradigm shift was inspired by how humans naturally perceive the world—as interacting objects with properties and behaviors.

Definition and Core Philosophy of OOP

Object Oriented Programming is a programming paradigm that organizes software design around objects rather than functions and logic. An object is a self-contained entity that contains both data (attributes or properties) and the operations (methods or behaviors) that can manipulate that data.

The fundamental philosophy of OOP rests on several key principles:

Real-World Modeling: OOP allows programmers to create software representations of real-world entities. For instance, if we are developing a university management system, we can create Student, Teacher, Course, and Department classes that mirror actual entities in a university.

Organization Through Objects: Instead of thinking about a program as a series of steps to be executed, OOP encourages thinking about a program as a collection of collaborating objects, each responsible for its own state and behavior.

Message Passing Paradigm: Objects communicate with each other by sending and receiving messages (function calls). When object A wants object B to perform an action, it sends a message to object B, which then responds appropriately.

Identifiers

Identifiers are names given to program elements such as variables, functions, classes, and objects. They serve as labels that programmers use to refer to these elements.

Rules for Forming Identifiers:

1. Identifiers must consist of letters (a-z, A-Z), digits (0-9), and underscores (_)
2. The first character must be a letter or underscore (cannot be a digit)
3. Identifiers cannot be C++ keywords (reserved words)
4. C++ is case-sensitive, meaning 'total' and 'Total' are different identifiers
5. While there is no theoretical limit on length, most compilers recognize at least the first 31 characters

Examples of Valid Identifiers:

```
cpp
studentName
_totalCount
maximumValue
x1
_private
```

Examples of Invalid Identifiers:

```
cpp
int 3rdPlace;       // Error: Begins with a digit
double my name;     // Error: Contains space
class int;          // Error: 'int' is a keyword
float first-name;   // Error: Hyphen not allowed
```

Best Practices for Naming Identifiers:

- Use meaningful names that indicate purpose (e.g., studentCount rather than sc)
- Follow consistent naming conventions (camelCase for variables, PascalCase for classes)
- Avoid names that differ only in case to prevent confusion
- Use underscores for constants and macro names

Keywords

Keywords, also called reserved words, are words that have special meaning in C++. They are part of the language syntax and cannot be used as identifiers for any purpose.

Common C++ keywords include:

Category	Keywords
Data Types	int, char, float, double, bool, void, short, long, signed, unsigned
Control Flow	if, else, switch, case, default, for, while, do, break, continue, goto
Classes	class, struct, union, public, private, protected, virtual, friend
Storage Class	auto, register, static, extern, mutable
Other	return, sizeof, typedef, namespace, using, new, delete, this, const

Constants in C++

Constants are fixed values that do not change during program execution. They provide a way to represent values that remain invariant.

Syntax: condition ? expression1 : expression2
If condition evaluates to true, the result is expression1; otherwise, the result is expression2.

```
cpp
int a = 10, b = 20;
int max = (a > b) ? a : b;        // max = 20

// Equivalent if-else:
if (a > b)
   max = a;
else
   max = b;
cpp
int x = 5;
cout << (x % 2 == 0 ? "Even" : "Odd");  // Output: "Odd"
```

Operator Precedence and Associativity

When an expression contains multiple operators, precedence determines the order of evaluation. Operators with higher precedence are evaluated before those with lower precedence.

Precedence Table (Highest to Lowest):

Level	Operators	Description
1	::	Scope resolution
2	() [] . -> ++ --	Postfix operators
3	++ -- + - ! ~ * &	Unary operators (prefix)
4	.* ->*	Pointer-to-member
5	* / %	Multiplicative
6	+ -	Additive
7	<< >>	Shift
8	< <= > >=	Relational
9	== !=	Equality
10	&	Bitwise AND
11	^	Bitwise XOR
12	|	Bitwise OR
13	&&	Logical AND
14	||	Logical OR
15	? :	Conditional
16	= += -= *= /= %=	Assignment
17	,	Comma

Associativity: When operators have the same precedence, associativity determines evaluation order (left-to-right or right-to-left).

```
cpp
int result = 5 + 3 * 4;          // 17 (not 32) - * has higher precedence
int result2 = (5 + 3) * 4;       // 32 - parentheses override precedence
int x = y = z = 10;              // Right-to-left associativity of assignment
```

String Operations

Assignment and Concatenation:

cpp

```
string firstName = "John";
string lastName = "Doe";
string fullName = firstName + " " + lastName;  // "John Doe"

firstName += "athan";         // "Jonathan"
```

Accessing Characters:

cpp

```
string str = "Hello";
char ch1 = str[0];            // 'H' (no bounds checking)
char ch2 = str.at(0);         // 'H' (with bounds checking)
str[1] = 'a';                 // "Hallo"
```

String Comparison:

cpp

```
string s1 = "apple";
string s2 = "banana";

if (s1 == s2) { /* equal */ }
if (s1 != s2) { /* not equal */ }
if (s1 < s2) { /* s1 comes before s2 lexicographically */ }
```

Important String Class Methods

Method	Description	Example
length() / size()	Returns string length	int len = str.length();
empty()	Checks if string is empty	if (str.empty())
clear()	Empties the string	str.clear();
append(str)	Appends a string	str.append(" world");
insert(pos, str)	Inserts at position	str.insert(5, "xxx");
erase(pos, len)	Erases characters	str.erase(5, 3);
replace(pos, len, str)	Replaces substring	str.replace(0, 5, "Hi");
find(str)	Finds substring	size_t pos = str.find("lo");
substr(pos, len)	Extracts substring	string sub = str.substr(0, 5);
c_str()	Returns C-string equivalent	const char* cstr = str.c_str();

Example:

cpp

```
#include <string>
#include <iostream>
using namespace std;

int main() {
    string text = "The quick brown fox jumps over the lazy dog";
```

```
    Complex c3 = c1 * c2;

    cout << "c1 * c2 = " << c3 << endl;

    return 0;
}
```

INHERITANCE

Introduction to Inheritance

Inheritance is a fundamental OOP concept that allows a class (derived class) to inherit properties and behaviors from another class (base class). It establishes an "is-a" relationship between classes.

Benefits of Inheritance:

- **Code Reusability:** Base class code can be used in multiple derived classes
- **Extensibility:** New features can be added by deriving from existing classes
- **Hierarchical Classification:** Represents real-world relationships naturally
- **Polymorphism Support:** Enables runtime polymorphism

Base and Derived Classes

Syntax for Inheritance:

cpp

```
class BaseClass {
    // Base class members
};

class DerivedClass : access_specifier BaseClass {
    // Derived class members
};
```

Example:

cpp

```
// Base class
class Vehicle {
protected:
    string brand;
    int year;

public:
    Vehicle(string b, int y) : brand(b), year(y) {}

    void start() {
        cout << "Vehicle starting..." << endl;
    }
```

```
}
inFile.close();
```

Important: Always check for EOF after a read operation, not before.

File Pointers and Random Access

File Pointer Functions

Function	Purpose
tellg()	Returns current position in input stream
tellp()	Returns current position in output stream
seekg(pos)	Moves input pointer to absolute position
seekp(pos)	Moves output pointer to absolute position
seekg(offset, dir)	Moves input pointer relative to dir
seekp(offset, dir)	Moves output pointer relative to dir

Seeking Directions

Direction	Description
ios::beg	From beginning of file
ios::cur	From current position
ios::end	From end of file

16.9.3 Random Access Examples

cpp

```
// Move to 10th byte from beginning
file.seekg(10, ios::beg);

// Move 5 bytes forward from current position
file.seekg(5, ios::cur);

// Move to end of file
file.seekg(0, ios::end);
```

Complete Random Access Example:

cpp

```
#include <fstream>
#include <iostream>
using namespace std;

int main() {
    // Create and write to file
    ofstream outFile("numbers.txt");
    for (int i = 1; i <= 10; i++) {
        outFile << i << " ";
    }
    outFile.close();

    // Read with random access
    ifstream inFile("numbers.txt");
```

```
    // Using algorithms
    sort(vec.begin(), vec.end());
    auto it = find(vec.begin(), vec.end(), 3);

    return 0;
}
```

Practice MCQs

1. Who developed C++ and at which laboratory?
a) Dennis Ritchie at Bell Laboratories
b) Bjarne Stroustrup at Bell Laboratories
c) James Gosling at Sun Microsystems
d) Guido van Rossum at CWI
Answer: Bjarne Stroustrup at Bell Laboratories

2. What are the four fundamental concepts or pillars of Object Oriented Programming?
a) Abstraction, encapsulation, inheritance, polymorphism
b) Classes, objects, methods, attributes
c) Data hiding, modularity, reusability, extensibility
d) Encapsulation, modularity, inheritance, overloading
Answer: Abstraction, encapsulation, inheritance, polymorphism

3. Which OOP concept is defined as the mechanism that binds together code and the data it manipulates, keeping both safe from outside interference and misuse?
a) Inheritance
b) Polymorphism
c) Encapsulation
d) Abstraction
Answer: Encapsulation

4. What is the name of the mechanism by which one class acquires the properties and behaviors of another class?
a) Encapsulation
b) Inheritance
c) Polymorphism
d) Abstraction
Answer: Inheritance

5. Which OOP concept allows objects of different types to respond to the same message (method call) in different ways?
a) Encapsulation
b) Inheritance
c) Polymorphism
d) Data hiding
Answer: Polymorphism

6. What is the mandatory entry point of every C++ program?
a) void main()
b) int main()
c) start()
d) begin()
Answer: int main()

7. Which symbol is used for single-line comments in C++?
a) /* ... */
b) #
c) //
d) --
Answer: //

8. What are the rules for forming valid identifiers in C++?
a) Can start with a digit, contain letters and underscores
b) Must consist of letters, digits, and underscores; first character must be a letter or underscore; cannot be a keyword
c) Can contain spaces and special characters
d) Must be written in uppercase letters only
Answer: Must consist of letters, digits, and underscores; first character must be a letter or underscore; cannot be a keyword

Software Engineering

INTRODUCTION

Software Engineering is one of the most fundamental subjects in computer science, forming the backbone of professional software development practices. These notes have been prepared specifically for BPSC Lecturer Computer Science subjective examination, providing comprehensive coverage of all concepts from basic to advanced levels.

FUNDAMENTALS OF SOFTWARE ENGINEERING

Introduction to Software Engineering

Definition and Basic Concepts

Software Engineering is defined as the systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software. This definition, adopted by the IEEE Standard Glossary of Software Engineering Terminology, emphasizes several key aspects that distinguish software engineering from simple programming.

The term "systematic" implies that software development follows a well-defined, organized, and methodical approach rather than ad-hoc or chaotic methods. Every activity from requirements gathering to deployment is performed according to established procedures and guidelines.

The word "disciplined" indicates that software engineers adhere to professional standards, best practices, and ethical guidelines throughout the development process. This discipline ensures consistency and reliability in software products.

"Quantifiable" means that software engineering emphasizes measurement and analysis. Progress, quality, costs, and other attributes are measured objectively, allowing for informed decision-making and continuous improvement.

"Development, operation, and maintenance" together constitute the complete software lifecycle. Development includes all activities from initial concept to delivery. Operation involves running the software in production environments. Maintenance encompasses all modifications made after delivery, including bug fixes, enhancements, and adaptations.

To understand software engineering fully, we must first understand what software itself is. Software is not merely computer programs; it includes three components:

1. Executable code that runs on computer hardware
2. Documentation that explains how to use and maintain the software
3. Data that the software processes and produces

Unlike hardware, software is intangible, easily replicated, and does not wear out physically. However, software deteriorates in a different sense through obsolescence, as requirements change and environments evolve.

The Software Crisis: Historical Background

The emergence of software engineering as a discipline can be traced directly to what is known as the "Software Crisis" of the 1960s and 1970s. During this period, as computers became more powerful and widespread, organizations began attempting to build larger and more complex software systems. These efforts frequently ended in failure, revealing fundamental problems with existing development approaches.

Practice MCQs

1. What is the formal definition of Software Engineering according to the IEEE Standard Glossary of Software Engineering Terminology?
a) The art of writing computer programs
b) The systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software
c) The study of algorithms and data structures
d) The process of testing software for bugs
Answer: The systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software

2. What are the three components that constitute software?
a) Programs, data, and users
b) Executable code, documentation, and data
c) Source code, object code, and libraries
d) Requirements, design, and implementation
Answer: Executable code, documentation, and data

3. What historical event led to the emergence of software engineering as a discipline?
a) The invention of the microprocessor
b) The Software Crisis of the 1960s and 1970s
c) The development of the Internet
d) The creation of the C programming language
Answer: The Software Crisis of the 1960s and 1970s

4. At which conference was the term "Software Engineering" first coined?
a) The IEEE Computer Conference 1970
b) The NATO Science Committee Conference in 1968
c) The ACM Computing Conference 1965
d) The IBM Developers Conference 1962
Answer: The NATO Science Committee Conference in 1968

5. What percentage of total software lifecycle costs is typically consumed by maintenance?
a) 20 to 30 percent
b) 40 to 50 percent
c) 60 to 80 percent
d) 80 to 90 percent
Answer: 60 to 80 percent

6. Which characteristic of software means that it cannot be seen or touched, making progress difficult to measure?
a) Malleability
b) Complexity
c) Intangibility
d) Replicability
Answer: Intangibility

7. What is the Software Development Life Cycle (SDLC)?
a) A programming language for developing software
b) A conceptual framework that describes the stages involved in software development from initial concept through final retirement
c) A testing methodology for finding bugs
d) A tool for managing software projects
Answer: A conceptual framework that describes the stages involved in software development from initial concept through final retirement

8. Which SDLC model is the oldest and most straightforward, proceeding through phases in strict sequence with each phase completed before the next begins?
a) Spiral model
b) Agile model
c) Waterfall model
d) Iterative model
Answer: Waterfall model

9. Who first formally described the Waterfall model in 1970?
a) Barry Boehm
b) Winston Royce
c) Kent Beck
d) Fred Brooks

Operating Systems

FUNDAMENTALS OF SOFTWARE AND OPERATING SYSTEMS

UNDERSTANDING SOFTWARE

Introduction to Software

Software represents the logical, intangible component of a computer system that brings hardware to life. Without software, even the most sophisticated and advanced computer hardware remains nothing more than an inert collection of electronic circuits, semiconductor chips, and interconnected components. The term "software" encompasses all the instructions, data, and programs that direct the operations of computer hardware.

When we examine a computer system from a holistic perspective, we recognize that it comprises two fundamental components: hardware and software. Hardware provides the physical infrastructure—the tangible elements that we can see and touch, such as the processor, memory modules, storage devices, input devices, and output devices. Software, in contrast, exists as patterns of bits, electromagnetic states, and logical constructs that instruct the hardware on what operations to perform.

The relationship between hardware and software is symbiotic. Hardware without software is incapable of performing any useful function, much like a human body without consciousness. Conversely, software without hardware has no physical medium to execute upon and remains merely an abstract concept. The seamless integration of these two components creates the powerful information processing systems that have transformed every aspect of modern life.

Formal Definition of Software

From an academic perspective, software can be formally defined as a comprehensive collection of computer programs, procedures, rules, and associated documentation that instructs a computer system on what tasks to perform and how to perform them. This definition encompasses several critical elements that merit detailed examination:

Computer Programs: These are sequences of instructions written in programming languages that specify operations to be performed by the computer. Programs are the executable components that directly control hardware actions.

Procedures: These are the established methods and protocols for using the software effectively. Procedures include operational guidelines, error handling approaches, and standard practices that ensure consistent and correct software utilization.

Rules: These define the constraints, permissions, and operational boundaries within which the software must function. Rules govern aspects such as data access, resource utilization, and interaction patterns.

Associated Documentation: This includes user manuals, technical reference materials, system documentation, and help files that explain how to install, operate, and maintain the software.

Instruction Function: The core purpose of software is to instruct the computer. These instructions range from low-level machine code operations to high-level application functions.

Intermediary Role: Software serves as the critical intermediary between human intentions expressed through commands and machine execution manifested through hardware operations.

Task Automation: Applications automate repetitive or complex tasks, saving time and reducing errors compared to manual methods.
Capability Extension: Applications extend human capabilities in areas such as calculation speed, information storage, visual representation, and communication reach.
Domain Expertise Embodiment: Applications encapsulate domain knowledge, making specialized techniques accessible to non-experts.
Creative Expression: Applications provide tools for artistic creation across multiple media types.
Information Management: Applications help organize, store, retrieve, and analyze information effectively.
Communication Enablement: Applications facilitate communication and collaboration across distances and time zones.
Entertainment Provision: Applications deliver engaging experiences for leisure and enjoyment.

Comparative Analysis of System and Application Software

Detailed Comparison Table

Aspect	System Software	Application Software
Primary Purpose	Manage hardware resources and provide platform for other software	Help users perform specific tasks
Control Relationship	Controls hardware directly	Controlled by users
Necessity	Essential - computer cannot run without it	Optional - installed based on user needs
Installation Timing	Usually pre-installed with hardware	Installed by users as needed
Users	Used by computer system itself	Used by end-users
Interaction Level	Low-level interaction with hardware	High-level interaction through GUI
Execution Mode	Typically runs in background	Runs in foreground with user interaction
Number of Instances	Usually one system software instance	Many application instances possible
Hardware Knowledge	Requires detailed hardware knowledge	Hardware details abstracted away
Programming Language	Often written in low-level languages (C, assembly)	Written in various high-level languages
Development Complexity	Extremely complex, long development cycles	Variable complexity based on purpose
Error Impact	Errors can crash entire system	Errors typically affect only the application
Update Frequency	Less frequent, major changes	Frequent updates and new versions
User Choice	Limited choice (OS tied to hardware)	Wide choice among competing products

2.4.2 Practical Analogy: Theater Production

To understand the relationship between system and application software, consider the analogy of a theater production:

System Software corresponds to the stage crew, lighting technicians, sound engineers, and production staff who work behind the scenes. These individuals:

- Set up and manage the physical infrastructure
- Ensure lighting and sound equipment functions properly
- Coordinate scene changes and special effects

Key Advances in PC Operating Systems:

Graphical User Interface (GUI):

- Windows, icons, menus, pointing device
- Consistent look and feel
- Reduced learning curve
- Visual file management

Multitasking:

- Multiple applications simultaneously
- Background processing
- Responsive user interface

Plug and Play:

- Automatic hardware detection
- Driver installation without user intervention
- Hot-swapping support

Networking Integration:

- Built-in TCP/IP stack
- Network file sharing
- Internet connectivity

Security Features:

- User accounts and permissions
- Firewall integration
- Automatic updates
- Encrypted file systems

Mobile and Embedded Operating Systems (1990s-Present)

Mobile Operating Systems:

The proliferation of smartphones and tablets created new OS requirements:

- Touch-based interfaces
- Power efficiency for battery operation
- Small form factor
- Integrated sensors (GPS, accelerometer, camera)
- App-based software model

Major Mobile OS:

- **iOS:** Apple's mobile OS for iPhone, iPad
- **Android:** Google's open-source mobile OS
- **Windows Phone:** Microsoft's mobile offering (discontinued)
- **Symbian:** Former leader (Nokia)
- **BlackBerry OS:** For BlackBerry devices

Characteristics:

- Touch-optimized interfaces
- App stores for software distribution
- Strict security models (sandboxing)
- Power management critical

- **Proportional Share:** Based on weights/importance
- **Deadline-Based:** Meet timing requirements

Accounting

The OS tracks resource usage for various purposes.

Usage Tracking:

The OS maintains counters and logs for:

- **CPU Time:** Per process, per user
- **Memory Usage:** Peak and average
- **Disk Access:** Reads, writes, bytes transferred
- **Network Usage:** Packets, bytes transmitted/received
- **Device Utilization:** Time devices active

Purposes of Accounting:

Billing:

- Charge users for resource consumption
- Allocate costs to projects/departments
- Support time-sharing service fees

Performance Tuning:

- Identify bottlenecks
- Detect unusual usage patterns
- Plan capacity upgrades
- Optimize resource allocation

Security Monitoring:

- Detect anomalous behavior
- Identify potential attacks
- Support forensic analysis
- Maintain audit trails

Planning:

- Forecast future resource needs
- Plan hardware acquisitions
- Schedule maintenance
- Allocate budgets

Protection and Security

The OS protects system resources and user data from unauthorized access and malicious activities.

Authentication:

Verify user identity before granting access:

- **Passwords:** Secret knowledge
- **Biometrics:** Fingerprint, face, iris recognition
- **Tokens:** Smart cards, security keys
- **Multi-Factor:** Combination of methods

Access Control:

Determine what authenticated users can do:

- **Discretionary Access Control (DAC):** Owners set permissions

The layered approach organizes the operating system into hierarchical layers, each built on the services provided by lower layers. Each layer has well-defined interfaces and interacts only with adjacent layers. This creates modularity and reduces complexity.

The THE System:

The layered approach was pioneered by Edsger Dijkstra in the THE (Technische Hogeschool Eindhoven) operating system. THE had six layers:

Layer	Function
Layer 5	User programs
Layer 4	I/O management
Layer 3	Operator console
Layer 2	Memory management
Layer 1	CPU scheduling and multiprogramming
Layer 0	Hardware

Typical Layered Organization:

A more general layered structure might be:

Layer 6: User Interface:

Command interpreters, graphical shells, application interfaces.

Layer 5: System Services:

File systems, networking protocols, security services.

Layer 4: Process Management:

Process scheduling, inter-process communication, synchronization.

Layer 3: Memory Management:

Virtual memory, paging, segmentation.

Layer 2: I/O Management:

Device drivers, interrupt handlers, buffering.

Layer 1: Hardware Abstraction:

Low-level hardware interfaces, interrupt vectors, timer management.

Layer 0: Hardware:

Physical processors, memory, devices.

Detailed Characteristics:

Hierarchical Dependence:

Each layer depends only on lower layers. Layer n can call functions in layer n-1 but not in layer n+1. This prevents circular dependencies and simplifies reasoning about the system.

Abstraction Boundaries:

Each layer hides implementation details of lower layers and provides abstracted services to higher layers. A programmer working at layer 4 need not know details of layer 2.

Well-Defined Interfaces:

Interfaces between layers are precisely specified and maintained. This allows independent development of layers as long as interfaces are respected.

Advantages in Detail:

Module Interaction:

Modules can interfere with each other through shared kernel data structures.

5.5 Summary of Structural Approaches

Approach	Key Characteristics	Advantages	Disadvantages	Examples
Monolithic	All kernel code in one address space	High performance, simple design	Poor reliability, hard to maintain	Early UNIX, MS-DOS
Layered	Hierarchical layers with well-defined interfaces	Modular, easy to debug	Performance overhead, rigid	THE, MULTICS
Microkernel	Minimal kernel, services as user processes	Reliable, extensible, portable	Performance overhead	Mach, QNX, MINIX
Modular	Core kernel with loadable modules	Performance + flexibility	Less isolation than microkernel	Linux, Windows, Solaris

Modern Practice:

Most contemporary operating systems use hybrid approaches:

- **Linux:** Monolithic kernel with extensive modularity
- **Windows:** Modular kernel with some microkernel influences
- **macOS:** XNU kernel combining Mach microkernel and BSD monolithic elements
- **Solaris:** Highly modular with dynamic loading

TYPES OF OPERATING SYSTEMS BASED ON USER INTERFACE

The user interface represents the point of interaction between users and the computer system. The evolution of user interfaces reflects changing priorities in computing—from maximizing efficiency for experts to maximizing accessibility for novices.

Command Line Interface (CLI)

Definition and Concept

A Command Line Interface is a text-based interface where users interact with the computer by typing specific commands. The system displays a prompt, the user types a command, and the system executes it and displays results. This interaction pattern is known as the read-evaluate-print loop (REPL).

The Command Interpreter (Shell):

The program that accepts, interprets, and executes commands is called a shell. The shell:

- Displays the prompt
- Reads user input
- Parses command line into command and arguments
- Locates executable program
- Creates process to run command
- Waits for completion (typically)
- Displays results and next prompt

Detailed Characteristics

Interaction Model:

The interaction is purely textual and linear. Users type commands, see output, and type more commands. There is no visual context beyond text.

Round Robin:
Simplest algorithm. Each process gets a fixed time quantum in circular order. Fair and predictable but doesn't prioritize interactive processes.
Priority Scheduling:
Processes have priorities. Higher priority processes run first. May cause starvation of low-priority processes.
Multilevel Queue:
Processes divided into classes (interactive, batch, system). Each class has its own scheduling algorithm.
Multilevel Feedback Queue:
Processes can move between queues based on behavior. Interactive processes get higher priority; CPU-bound processes move to lower priority queues.
Key Metrics
Response Time:
Time from submission of request to first response. Critical for interactive satisfaction.
Turnaround Time:
Time from job submission to completion. Important for batch work.
Waiting Time:
Total time process spends in ready queue. Affects fairness perception.
Throughput:
Number of jobs completed per time unit. Measures system productivity.
Examples
All modern desktop and server operating systems provide time-sharing/multitasking:

- Microsoft Windows
- macOS
- Linux
- UNIX variants

Real-Time Operating Systems (RTOS)

Definition and Concept
Real-time operating systems are designed for systems requiring immediate, guaranteed response to events. Correctness depends not only on logical results but also on the timeliness of production.
Fundamental Principle:
In real-time systems, producing the correct result too late is equivalent to producing an incorrect result. A flight control system that calculates the correct correction after the plane has crashed has failed completely.
Classification of Real-Time Systems
Hard Real-Time Systems:
Definition: Missing a deadline has catastrophic consequences. The system must guarantee response within specified time constraints under all conditions.
Characteristics:

- Deterministic behavior required
- Worst-case execution time must be calculated and guaranteed
- No virtual memory (paging introduces unpredictability)

Essential Keyboard Shortcuts:

Shortcut	Function
Ctrl+C	Copy selected
Ctrl+X	Cut selected
Ctrl+V	Paste
Ctrl+Z	Undo
Ctrl+Y	Redo
Ctrl+A	Select all
Ctrl+S	Save
Ctrl+P	
Ctrl+F	Find/Search
Alt+Tab	Switch between open windows
Alt+F4	Close current window
Windows+D	Show desktop
Windows+E	Open File Explorer
Windows+L	Lock computer
Windows+R	Open Run dialog
Print Screen	Capture entire screen
Alt+Print Screen	Capture active window

File and Folder Management

File Explorer

File Explorer (formerly Windows Explorer) is the primary tool for browsing and managing files and folders.

File Explorer Components:

Navigation Pane:

Left side showing tree structure of:

- Quick Access (frequent folders)
- OneDrive
- This PC (drives and system folders)
- Network

Content Pane:

Main area showing contents of selected location. Can display as:

- Icons (extra large to small)
- List
- Details (with columns for name, date, size, type)
- Tiles
- Content (thumbnails with details)

Address Bar:

Shows current path. Click segments to navigate up hierarchy. Type path directly to go to location.

Search Box:

Search current folder and subfolders by filename or content.

Ribbon/Toolbar:

Commands for file operations, view options, sharing.

Multitasking vs. Multiprogramming:

Aspect	Multiprogramming	Multitasking
Primary Goal	Maximize CPU utilization	Provide interactive response
Switching Trigger	I/O wait (voluntary)	Timer (preemptive)
User Interaction	None	Continuous
Focus	Throughput	Response time
Scheduling	Simple	Complex
Historical Era	1960s	1970s onward

Multithreading

Definition:

Multithreading divides a single process into multiple threads that can execute concurrently, sharing the process's resources.

Goal:

Improve application performance and responsiveness by parallelizing within a process.

Key Insight:

Many applications have multiple independent tasks that can execute simultaneously while sharing data.

Characteristics:

- Multiple execution paths within one process
- Shared memory and resources
- Lightweight compared to processes
- Can run on multiple processors
- Requires synchronization
- Application-level parallelism

Comparison Summary:

Aspect	Multiprogramming	Multitasking	Multithreading
Unit	Programs/processes	Tasks/processes	Threads
Level	System-wide	System-wide	Process-specific
Parallelism	Overlap I/O and computation	Illusion of simultaneity	True or apparent parallelism
Resource Sharing	Separate address spaces	Separate address spaces	Shared address space
Purpose	Utilize CPU during I/O	Interactive computing	Application concurrency
Management	OS	OS	OS or user-level library

Key Features of Modern Operating Systems

Modern operating systems incorporate numerous features that enhance usability, performance, reliability, and security.

Multitasking

All modern general-purpose OS support multitasking, allowing multiple applications to run concurrently. Features include:

- Preemptive scheduling
- Priority-based execution
- Background processing
- Responsive foreground interaction

Practice MCQs

1. What is the formal definition of software according to academic perspective?
a) Computer programs and data only
b) A comprehensive collection of computer programs, procedures, rules, and associated documentation that instructs a computer system on what tasks to perform and how to perform them
c) The physical components of a computer system
d) The user interface of an operating system
Answer: A comprehensive collection of computer programs, procedures, rules, and associated documentation that instructs a computer system on what tasks to perform and how to perform them

2. What are the two fundamental categories of software?
a) System software and utility software
b) Application software and utility software
c) System software and application software
d) Operating systems and device drivers
Answer: System software and application software

3. What is the primary purpose of system software?
a) To help users perform specific tasks
b) To manage and operate computer hardware, providing a stable platform for application software
c) To provide entertainment
d) To create documents and spreadsheets
Answer: To manage and operate computer hardware, providing a stable platform for application software

4. Which of the following are major components of system software?
a) Word processors, spreadsheets, presentations
b) Operating systems, device drivers, utility programs, language translators
c) Web browsers, email clients, games
d) Database software, project management tools
Answer: Operating systems, device drivers, utility programs, language translators

5. What is the most critical and comprehensive component of system software that acts as the master controller of the entire computer system?
a) Device driver
b) Utility program
c) Operating system
d) Language translator
Answer: Operating system

6. What are the five generations of software evolution in correct chronological order?
a) Assembly Language Era, Machine Language Era, Operating Systems Era, Personal Computing Era, Networked Era
b) Machine Language Era, Assembly Language and Early High-Level Languages Era, Operating Systems and System Software Era, Personal Computing and Software Industry Era, Networked and Mobile Software Era
c) Personal Computing Era, Machine Language Era, Networked Era, Operating Systems Era, Assembly Language Era
d) Networked Era, Personal Computing Era, Operating Systems Era, Assembly Language Era, Machine Language Era
Answer: Machine Language Era, Assembly Language and Early High-Level Languages Era, Operating Systems and System Software Era, Personal Computing and Software Industry Era, Networked and Mobile Software Era

7. What are the distinctive characteristics of software that differentiate it from hardware?
a) Physical nature, manufacturing process, visibility
b) Logical rather than physical nature, development rather than manufacturing, complexity, changeability, invisibility, no

Discrete Structures

SET THEORY - THE FOUNDATION OF DISCRETE STRUCTURES

Introduction to Set Theory

The study of Discrete Structures begins with set theory, a fundamental concept that underpins virtually all of modern mathematics and computer science. Set theory was pioneered by the German mathematician Georg Cantor around 1870. Cantor defined a set as a collection of discrete, distinguishable objects that share a certain relationship or property . This seemingly simple idea has profound implications for computer science, as sets provide the mathematical basis for everything from database theory to programming language design.

In the context of discrete structures, we study sets because they allow us to reason about collections of objects in a precise and formal manner. When we write computer programs, we constantly work with collections of data—lists of names, groups of records, collections of files—and set theory gives us the language and tools to describe and manipulate these collections mathematically.

Defining Sets: Terminology and Notation

Basic Definitions

A **set** is an unordered collection of distinct objects, called **elements** or **members** of the set. The objects in a set can be anything: numbers, letters, people, other sets, or abstract concepts. What makes a set a set is that we can unambiguously determine whether any given object belongs to it or not.

The **membership** of an object in a set is denoted by the symbol ∈. If x is an element of set A, we write x ∈ A. If x is not an element of A, we write x ∉ A. This binary relationship—an object is either entirely in a set or entirely not in it—is fundamental to classical set theory .

It is important to distinguish between a set and a **collection**. In some contexts, particularly in computer science, the term "collection" may allow duplicate elements, whereas a set, by definition, contains each element only once. For example, the group of students in a classroom is a set because each student is a distinct individual. However, a shopping list that includes "milk" twice is technically a collection (or multiset) rather than a set .

Methods of Describing Sets

There are two primary ways to describe a set: the roster method and the set-builder notation.

The Roster Method (Tabular Form): This method simply lists all the elements of the set within curly braces, separated by commas. For example:

- A = {1, 2, 3, 4, 5}
- B = {a, e, i, o, u}
- C = {red, green, blue}

When the set has many elements, we can use ellipses to indicate a pattern. For instance:

- D = {1, 2, 3, ..., 100} represents the set of natural numbers from 1 to 100.
- E = {2, 4, 6, 8, ...} represents the set of all positive even numbers.

Set-Builder Notation: This method describes the set by stating the property or properties that its members must satisfy. The general form is {x | P(x)}, which is read as "the set of all x such that P(x) is true." The vertical bar "|" means "such that." Sometimes a colon ":" is used instead.

Disjunction (∨):

p	q	p ∨ q
T	T	T
T	F	T
F	T	T
F	F	F

Disjunction is true when at least one component is true. This is inclusive or—the "or both" case is included. In everyday language, "or" is sometimes exclusive ("You can have coffee or tea"—usually meaning not both). In logic, ∨ always means inclusive unless explicitly stated otherwise.

Conditional (→):

p	q	p → q
T	T	T
T	F	F
F	T	T
F	F	T

The conditional is the most subtle connective. p → q is false only when p is true and q is false. In all other cases, it is true. This definition may seem counterintuitive, but it reflects mathematical practice: an implication is considered true unless the hypothesis is true and the conclusion false.

In p → q, p is called the **hypothesis** (or antecedent or premise), and q is called the **conclusion** (or consequent).

Biconditional (↔):

p	q	p ↔ q
T	T	T
T	F	F
F	T	F
F	F	T

The biconditional is true when p and q have the same truth value. It is equivalent to (p → q) ∧ (q → p).

Precedence of Logical Operators

To avoid ambiguity in compound propositions, we need rules of precedence, just as in arithmetic. The standard precedence order (from highest to lowest) is:

1. Negation (¬)
2. Conjunction (∧)
3. Disjunction (∨)
4. Conditional (→)
5. Biconditional (↔)

Parentheses can and should be used to override precedence or to improve readability.

Example: ¬p ∧ q → r is interpreted as ((¬p) ∧ q) → r, not ¬(p ∧ (q → r)) or any other grouping.

Logical Equivalences

Two compound propositions are **logically equivalent** if they have the same truth value for all possible truth values of their component propositions. Logical equivalence is denoted by the symbol ≡ or ⇔.

Important Logical Equivalences

Identity Laws:

- p ∧ T ≡ p

Practice MCQs

1. Who is credited with pioneering set theory around 1870?
a) John Venn
b) Georg Cantor
c) Augustus De Morgan
d) Gottlob Frege
Answer: Georg Cantor

2. What is the formal definition of a set?
a) A collection of objects that can be listed in any order
b) An unordered collection of distinct objects called elements or members of the set
c) A collection of elements that must be arranged in alphabetical order
d) A group of objects that allows duplicate entries
Answer: An unordered collection of distinct objects called elements or members of the set

3. What are the two primary methods for describing sets?
a) Listing method and diagram method
b) Roster method (tabular form) and set-builder notation
c) Numerical method and alphabetical method
d) Finite method and infinite method
Answer: Roster method (tabular form) and set-builder notation

4. What is the roster method for describing sets?
a) Describing the set by stating the property its members must satisfy
b) Listing all the elements of the set within curly braces, separated by commas
c) Drawing a diagram with circles representing the set
d) Using a mathematical formula to define the set
Answer: Listing all the elements of the set within curly braces, separated by commas

5. What is set-builder notation?
a) Listing all elements within curly braces
b) Describing the set by stating the property or properties that its members must satisfy, written as {x | P(x)}
c) Using ellipses to indicate patterns in a set
d) Drawing Venn diagrams to represent sets
Answer: Describing the set by stating the property or properties that its members must satisfy, written as {x | P(x)}

6. What is the symbol for the set of natural numbers?
a) Z
b) Q
c) N
d) R
Answer: N

7. What is the symbol for the set of integers?
a) N
b) Z
c) Q
d) R
Answer: Z

8. What is the symbol for the set of rational numbers?
a) R
b) C
c) Q
d) Z
Answer: Q

9. What is the symbol for the empty set?
a) {0}
b) U
c) ∅ or {}
d) E
Answer: ∅ or {}

10. What is the cardinality of a set?
a) The number of elements in the set, denoted |A| or n(A)
b) The total sum of all elements in the set
c) The average value of elements in the set
d) The type of elements in the set
Answer: The number of elements in the set,

Data Structures And Algorithms

INTRODUCTION TO DATA STRUCTURES

Understanding Data and Information

Before delving into data structures, it is essential to understand what data means in the context of computer science. **Data** refers to raw, unorganized facts, figures, or symbols that represent information about entities, events, or concepts in the real world. Data by itself carries no inherent meaning or context. For example, consider the sequence of numbers "25, 170, 75". These values are merely abstract symbols until we assign them meaning within a specific context.

When we provide context to data, it transforms into **information**. For instance, if we interpret the numbers as "age = 25 years, height = 170 centimeters, weight = 75 kilograms", they become meaningful information that can be used for decision-making and analysis. This transformation from raw data to meaningful information is at the heart of computing.

In computer systems, data exists at multiple levels of abstraction. At the lowest level, all data is stored as binary digits or **bits** (0s and 1s). However, programming languages and computer systems provide higher-level abstractions such as integers, characters, floating-point numbers, and more complex structures that make data manipulation practical and meaningful for human programmers.

Definition and Concept of Data Structures

A **data structure** is a specialized format for organizing, processing, retrieving, and storing data in a computer's memory or storage system. More formally, a data structure is a systematic way of organizing data elements along with a set of operations that can be performed on that data, designed in such a manner that operations on the data can be performed efficiently.

The concept of a data structure encompasses two fundamental aspects:

First, there is the **organization aspect** which defines how data elements relate to each other and how they are arranged in memory. This organization determines the logical relationships between data elements, such as sequential ordering, hierarchical relationships, or arbitrary connections.

Second, there is the **operational aspect** which defines what operations can be performed on the organized data and how efficiently these operations can be executed. Common operations include insertion of new elements, deletion of existing elements, searching for specific elements, traversing through all elements, and sorting the elements according to some criterion.

To understand data structures intuitively, consider an analogy with a library. A library contains thousands of books (the data). How these books are organized on shelves determines how quickly a librarian can find a specific book. Books could be arranged by genre (creating categorical organization), by author's name (alphabetical organization), by publication date (temporal organization), or by any other scheme. The chosen organization method is analogous to a data structure. Similarly, the procedures for finding, adding, or removing books correspond to the operations supported by the data structure.

Why Data Structures Are Essential

Data structures form the backbone of computer science and software development for several compelling reasons:

Efficiency in Operations: The choice of data structure directly impacts how efficiently operations can be performed. For example, searching for an element in a collection of one million items might take only

```
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;  // Avoid overflow

        if (arr[mid] == key) {
            return mid;  // Found
        }
        else if (arr[mid] < key) {
            left = mid + 1;  // Search in right half
        }
        else {
            right = mid - 1;  // Search in left half
        }
    }

    return -1;  // Not found
}
```

Time Complexity: O(log n) in all cases (best, average, worst). Each comparison eliminates half of the remaining elements.

Comparison of Linear and Binary Search:

Feature	Linear Search	Binary Search
Requirement	None	Array must be sorted
Time Complexity	O(n)	O(log n)
Implementation	Simple	More complex
Small Arrays	Efficient	Overkill
Large Arrays	Impractical	Highly efficient

Special Types of Arrays

Sparse Arrays

A **sparse array** is an array in which most elements have the same value (typically zero). Storing all elements in a standard array would waste memory. Sparse arrays use special representations to store only non-default values.

Example: A 1000×1000 matrix with only 50 non-zero elements. Storing as a standard array would require 1,000,000 elements, but the sparse representation needs only 150 values (each non-zero element stored as a triple: row, column, value).

Common Representations:

Coordinate List (COO):

Store each non-zero element as a triple (row, column, value):

text

Original 5×5 matrix:

0 0 0 0 0

0 0 8 0 0

- Can be traversed in both directions (forward and backward)
- Deletion is more efficient when a node pointer is given (no need to traverse from head to find predecessor)
- Can implement operations like "move backward" easily

Disadvantages:

- Extra memory for the previous pointer (more overhead)
- More complex operations (must update two pointers for each insertion/deletion)
- Slightly slower operations due to maintaining two pointers

Basic Operations:

Insertion at Beginning:

c

```
struct DNode* insertAtBeginning(struct DNode* head, int newData) {
    struct DNode* newNode = (struct DNode*)malloc(sizeof(struct DNode));
    newNode->data = newData;
    newNode->prev = NULL;
    newNode->next = head;

    if (head != NULL) {
        head->prev = newNode;
    }

    return newNode;
}
```

Insertion at End:

c

```
struct DNode* insertAtEnd(struct DNode* head, int newData) {
    struct DNode* newNode = (struct DNode*)malloc(sizeof(struct DNode));
    newNode->data = newData;
    newNode->next = NULL;

    if (head == NULL) {
        newNode->prev = NULL;
        return newNode;
    }

    struct DNode* current = head;
    while (current->next != NULL) {
        current = current->next;
    }

    current->next = newNode;
    newNode->prev = current;
```

```
    else if (key > root->data)
        root->right = deleteNode(root->right, key);
    else {
        // Node found - now handle the three cases

        // Case 1: No child or Case 2: One child
        if (root->left == NULL) {
            struct TreeNode* temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL) {
            struct TreeNode* temp = root->left;
            free(root);
            return temp;
        }

        // Case 3: Two children
        // Find inorder successor (smallest in right subtree)
        struct TreeNode* temp = minValueNode(root->right);

        // Copy the inorder successor's value to this node
        root->data = temp->data;

        // Delete the inorder successor
        root->right = deleteNode(root->right, temp->data);
    }
    return root;
}
```

Time Complexity: O(h)

Example - Delete node with two children:

text

Delete 50:

```
   50                    60
  / \                   / \
 30   70       →        30   70
/ \   / \              / \    \
20 40 60 80            20 40   80
```

(Using inorder successor 60)

BST Performance Issues

```
    }

    int e = 0;  // Index for result
    int i = 0;  // Index for sorted edges

    while (e < V-1 && i < E) {
        struct Edge nextEdge = edges[i++];

        int x = find(parent, nextEdge.src);
        int y = find(parent, nextEdge.dest);

        // If including this edge doesn't cause cycle
        if (x != y) {
            result[e++] = nextEdge;
            unionSets(parent, x, y);
        }
        // Else discard the edge
    }

    // Print the MST
    printf("Edge \tWeight\n");
    for (i = 0; i < e; i++) {
        printf("%d - %d \t%d\n", result[i].src, result[i].dest, result[i].weight);
    }
}
```

Time Complexity: O(E log E) = O(E log V) (since sorting dominates)

Comparison of MST Algorithms

Feature	Prim's	Kruskal's
Approach	Vertex-based	Edge-based
Data Structure	Priority queue (min-heap)	Union-Find
Best for	Dense graphs	Sparse graphs
Time Complexity	$O(V^2)$ or O(E log V)	O(E log E)
Implementation	Simpler with matrix	Requires Union-Find

Shortest Path Algorithms

Dijkstra's Algorithm

Dijkstra's algorithm finds the shortest paths from a source vertex to all other vertices in a weighted graph with **non-negative** edge weights.

Algorithm:

text

1. Initialize distances: dist[source] = 0, all others = INFINITY
2. Create a set of unvisited vertices
3. While unvisited set is not empty:

Minimum spanning tree (dense)	Prim's (adjacency matrix)
Minimum spanning tree (sparse)	Kruskal's
Graph traversal	DFS (deep exploration), BFS (shortest path)
Topological sort	Kahn's or DFS-based

Practice MCQs

1. What is data in the context of computer science?
a) Processed information that has meaning and context
b) Raw, unorganized facts, figures, or symbols that represent information about entities, events, or concepts in the real world
c) The output of a computer program
d) Binary digits stored in memory
Answer: Raw, unorganized facts, figures, or symbols that represent information about entities, events, or concepts in the real world

2. What is the formal definition of a data structure?
a) A collection of algorithms for processing data
b) A specialized format for organizing, processing, retrieving, and storing data in a computer's memory or storage system
c) A programming language construct
d) A type of computer memory
Answer: A specialized format for organizing, processing, retrieving, and storing data in a computer's memory or storage system

3. What are the two fundamental aspects encompassed by the concept of a data structure?
a) Storage and retrieval
b) Organization aspect and operational aspect
c) Input and output
d) Compile-time and run-time
Answer: Organization aspect and operational aspect

4. What is an Abstract Data Type (ADT)?
a) A concrete implementation of a data structure in a programming language
b) A mathematical model of a data type, specified in terms of the data it can hold and the operations that can be performed on that data
c) A primitive data type provided by programming languages
d) A class in object-oriented programming
Answer: A mathematical model of a data type, specified in terms of the data it can hold and the operations that can be performed on that data

5. What is the difference between an Abstract Data Type (ADT) and a Data Structure?
a) ADT is for integers; Data Structure is for strings
b) ADT specifies what operations are possible; Data Structure specifies how operations are implemented
c) ADT is static; Data Structure is dynamic
d) There is no difference
Answer: ADT specifies what operations are possible; Data Structure specifies how operations are implemented

6. What is the time complexity of accessing an element in an array by index?
a) O(n)
b) O(log n)
c) O(1)
d) $O(n^2)$
Answer: O(1)

7. What is the defining characteristic of arrays in terms of memory allocation?
a) Elements are stored in linked nodes
b) Elements are stored in contiguous memory locations
c) Elements are stored in random memory locations
d) Elements are stored in a hierarchical structure
Answer: Elements are stored in contiguous memory locations

Database Systems

Chapter 1: Introduction to Databases and Data Management

Understanding Data, Information, and Knowledge

The foundation of database systems lies in understanding the hierarchy of data processing. These three terms—data, information, and knowledge—represent distinct levels of value addition in the context of computing and decision-making.

Data consists of raw, unprocessed facts and figures that have no meaning by themselves. Data can take various forms including numbers, words, measurements, observations, images, or sounds. For example, the individual temperature readings recorded throughout a day, such as "35°C, 12:00 PM" or a list of student names and their obtained marks, constitute data. Data exists as isolated facts without context or interpretation. It is the lowest level of abstraction and serves as the raw material for all further processing. In a computer system, data functions as the input that is fed into various programs and processes. A crucial characteristic of data is that it is difficult to reproduce if lost, making it a valuable but vulnerable asset. Moreover, data exists as an independent entity, meaning it can exist without any interpretation or context applied to it.

Information is data that has been processed, organized, or structured to make it meaningful and useful. When raw data is placed in a context, processed through calculations, or organized in a particular way, it transforms into information. For instance, when daily temperature readings are analyzed to produce a weather forecast, or when student marks are processed to calculate percentages and assign grades, data becomes information. Information is valuable because it supports decision-making processes. Unlike raw data, information can be more easily reproduced—if a calculated result is lost, it can be recalculated from the original data. Information depends entirely on data; without data, there can be no information. In a computer system, information represents the output of data processing activities.

Knowledge represents the highest level of this hierarchy. Knowledge is the actionable insight gained from understanding patterns and relationships within information. When information is analyzed, interpreted, and combined with experience and context, it transforms into knowledge. For example, recognizing that students with consistent attendance patterns tend to perform better academically is knowledge derived from analyzing information about attendance and grades. Knowledge enables prediction, decision-making, and strategic planning. It represents the application of information to solve problems and make informed choices.

The transformation process follows a clear path: data becomes information through processing, and information becomes knowledge through analysis and interpretation. This progression is fundamental to understanding why organizations invest heavily in database systems—they need reliable data to produce meaningful information that ultimately generates actionable knowledge.

Evolution from File Systems to Databases

Before the advent of database management systems, organizations relied on File Management Systems (FMS) to store and manage their data. Understanding this evolutionary journey is essential because it explains why databases became necessary and what problems they were designed to solve.

File Management Systems represented the earliest approach to computerized data storage. In this model, data was stored in independent files, each containing collections of records, which themselves were

Mutually Exclusive Data occurs when attributes represent opposite conditions. For example, having both Married and Single flags is redundant—they are simply opposites. These should be combined into a single attribute like Marital_Status.

First Normal Form (1NF)

First Normal Form (1NF) is the most basic level of normalization. A table is in 1NF if it satisfies two conditions:

First, it must have a primary key that uniquely identifies each row. This is a fundamental requirement for any table.

Second, all attribute values must be **atomic**—that is, single-valued and indivisible. This means no repeating groups and no multi-valued attributes. Each cell in the table must contain exactly one value, not a list or set of values.

Consider a table that attempts to store student courses like this:

Student_ID	Name	Courses
101	Ali	CS101, MATH201
102	Sara	CS101, PHYS101, ENGL101

This table violates 1NF because the Courses column contains multiple values (repeating groups). To achieve 1NF, we must remove the repeating group by creating a separate row for each course:

Student_ID	Name	Course
101	Ali	CS101
101	Ali	MATH201
102	Sara	CS101
102	Sara	PHYS101
102	Sara	ENGL101

Now each cell contains exactly one value. However, note that we have introduced redundancy—the student name is repeated for each course. This redundancy will be addressed in higher normal forms.

Second Normal Form (2NF)

Second Normal Form (2NF) applies to tables with composite primary keys. A table is in 2NF if it meets two conditions: it must already be in 1NF, and every non-key attribute must be **fully functionally dependent** on the entire primary key.

The problem that 2NF addresses is **partial dependency**—when a non-key attribute depends on only part of a composite primary key. Consider our 1NF table with composite primary key {Student_ID, Course}. The non-key attribute Name depends only on Student_ID, not on the full combination of Student_ID and Course. This is a partial dependency.

Partial dependencies cause several problems. If we insert a new student who hasn't yet enrolled in any course, we cannot record their name because the primary key requires a course. This is an **insertion anomaly**. If we delete a student's last course, we might accidentally delete the student's name as well (deletion anomaly). And if a student changes their name, we must update multiple rows, risking inconsistency if we miss some (update anomaly).

To achieve 2NF, we decompose the table to remove partial dependencies. We create separate tables:

Practice MCQs

1. What is data in the context of computer systems?
a) Processed information that has meaning and context
b) Raw, unprocessed facts and figures that have no meaning by themselves
c) Actionable insights gained from understanding patterns
d) The output of database queries
Answer: Raw, unprocessed facts and figures that have no meaning by themselves

2. What is information?
a) Raw, unprocessed facts and figures
b) Data that has been processed, organized, or structured to make it meaningful and useful
c) Actionable insights from understanding patterns
d) The metadata stored in databases
Answer: Data that has been processed, organized, or structured to make it meaningful and useful

3. What is knowledge in the context of data processing?
a) Raw facts stored in databases
b) Processed data presented in reports
c) Actionable insight gained from understanding patterns and relationships within information
d) The structure of database tables
Answer: Actionable insight gained from understanding patterns and relationships within information

4. What were the major problems with File Management Systems (FMS) that led to the development of database systems?
a) Fast access, high security, easy maintenance
b) Data redundancy, data inconsistency, data-program dependence, data isolation, integrity problems, atomicity problems, poor security, intensive coding
c) Too many users, complex queries, high cost
d) Limited storage, slow processors, small memory
Answer: Data redundancy, data inconsistency, data-program dependence, data isolation, integrity problems, atomicity problems, poor security, intensive coding

5. What is the formal definition of a database?
a) A collection of files stored on a computer
b) A shared, integrated, and logically related collection of data, along with a description of that data (metadata)
c) A software program for data entry
d) A spreadsheet containing business information
Answer: A shared, integrated, and logically related collection of data, along with a description of that data (metadata)

6. What is a Database Management System (DBMS)?
a) A collection of data tables
b) A software system that enables users to create, maintain, manipulate, and access databases
c) A hardware device for storing data
d) A programming language for queries
Answer: A software system that enables users to create, maintain, manipulate, and access databases

7. What is program-data independence in the database approach?
a) The data is stored in programs
b) Changes to data structure do not necessarily require changes to the programs that access the data
c) Programs cannot access data directly
d) Data and programs are stored separately on different servers
Answer: Changes to data structure do not necessarily require changes to the programs that access the data

8. What is a database model?
a) A physical file structure