PRODUCT INDEX

INDEX

PROFESSORS ADDA

Trusted By Toppers

GET BEST SELLER HARD COPY NOTES

CLICK HERE TO GET

+91 7690022111+91 9216228788

UGC NET Commerce

Margdarshika Booklet

One-Stop Syllabus Coverage

Covers the entire Commerce UGC NET syllabus – 10 units from Business Environment to Tax Planning – all in one concise guide.

Unit-Wise 'What to Study' Focus

Each unit begins with a clear list of highly-focused topics, ensuring you spend time oly on what's most important.

How-to Study'-Strategies

Includes proven techniques – mind maps, case-study deep dives, comparative tables, and formula cheets – to make your revision an efficient

Exam-Oriented Tips

For every unit, you get MCQ-focused advice: high-yield areas, common question formats (match-the-following, true/false, scenario-baed), and quick shortcuts

Ready-Made Study Kit with Support

Complete study material -PDF booklet, structured guidelines, plus WhatsApp support (76900 22111 / 92162 28788) for anv dobts

UGC NET CSA UNIT-5

This guide outlines the key topics from Unit 5, "System Software and Operating System," and provides suggestions for studying them for the UGC NET exam. This unit covers the fundamental software that manages computer hardware and provides services to applications.

1. System Software

This foundational section introduces the concept of system software.

- **Definition:** Understand what system software is and its role as an interface between hardware and applications.
- **Examples:** Know common examples of system software (OS, BIOS, boot program, assembler, device driver, utilities, development tools).
- **Types:** Understand the important types of system software (Operating systems, Programming language translators, Communication Software, Utility programs).
- **Features:** Familiarize yourself with the characteristics of system software (closer to the system, low-level language, difficult to design, fast, less interactive, smaller size, hard to manipulate).

Study Tips:

- Clearly differentiate system software from application software.
- Understand the purpose of different types of system software.

2. Programming Languages and Translators

This section covers different levels of programming languages and how they are converted.

- **Machine Language:** Understand machine language as low-level binary code, specific to processor architecture.
- **Assembly Language:** Understand assembly language as a low-level language using mnemonics, translated by an assembler.
- **High Level Language:** Understand high-level languages as human-readable languages using English and mathematical symbols, portable, and easier to use. Know their advantages and drawback (poor hardware control).

- **Compiler:** Understand a compiler as a translator that converts the entire source code to machine code before execution. Know its steps (parsing, analysis, code generation, linking).
- **Interpreter:** Understand an interpreter as a translator that converts and executes code line-by-line during runtime.
- **Difference between Compiler and Interpreter:** This is a key comparison. Understand their differences in terms of output, process, speed, error reporting, memory requirement, best use cases, and code optimization.
- **Role:** Understand the specific roles of compilers and interpreters.
- **Object Code:** Understand object code as an intermediate code generated by compilers, often platform-independent before final linking/interpretation.
- **Java (Compiled and Interpreted):** Understand how Java uses both compilation (to bytecode/object code) and interpretation (by JVM).

Study Tips:

- Master the differences between machine, assembly, and high-level languages.
- Understand the trade-offs between using a compiler versus an interpreter.
- Know the process of translating a high-level language program into executable code.

3. Linking and Loading

This section covers the utilities that prepare programs for execution.

- **Linking:** Understand linking as combining object codes and libraries into an executable file.
- **Static Linking:** Understand static linking (performed before execution, copies library routines, larger executable, less runtime dependency).
- **Dynamic Linking:** Understand dynamic linking (performed at runtime, shares libraries, smaller executable, more runtime dependency, flexible updates). Know its advantages and disadvantages.
- **Loading:** Understand loading as bringing the executable file into main memory for execution.
- **Static Loading:** Understand static loading (entire program loaded before execution).

- **Dynamic Loading:** Understand dynamic loading (parts loaded on demand during execution).
- **Differences:** Understand the key differences between linking and loading, and between static and dynamic linking/loading.

Study Tips:

- Understand the purpose of linking and loading in the software development lifecycle.
- Know when static vs. dynamic linking/loading is preferred.

4. Macros and Debuggers

This section covers tools used in programming.

- **Macros:** Understand macros as sequences of instructions assigned a name for reuse in assembly language. Know their definition and invocation.
- **Debugger:** Understand a debugger as a software tool for testing and finding errors (bugs) in programs. Know its function (testing, finding errors, stepping through code, modifying state).

Study Tips:

- Understand how macros help with code modularity in assembly language.
- Know the basic purpose and features of a debugger.

5. Operating System (OS)

This is a major component of system software.

- **Definition:** Understand the OS as an interface between the user/applications and hardware, managing resources and providing services.
- **Why Learn OS:** Understand its fundamental role in managing computer resources.
- **Functions/Applications:** Know the important functions of an OS (Memory Management, Processor Management, Device Management, File Management, Security, Control over system performance, Job accounting, Error detecting aids, Coordination).
- **Structure:** Understand different OS structures (Simple, Layered). Focus on the layered approach (hiding details, defining interfaces).
- **Services:** Understand the services provided by the OS (Kernel operations, Command interpreter and utility services, Batch processing services, File and directory synchronization services).

- **Operations:** Understand the major operations managed by the OS (Process Management, Memory Management, Device Management, File Management).

Study Tips:

- Understand the core responsibilities of an operating system.
- Know the different ways an OS can be structured and the benefits of layering.
- Familiarize yourself with the key services provided by the OS.

6. System Calls

This section covers the interface between user programs and the OS kernel.

- **Concept:** Understand system calls as the programmatic way user programs request services from the OS kernel. Know that they provide an API.
- **Services Provided:** Know the categories of services accessed via system calls (Process creation/management, Main memory management, File Access/management, Device handling, Protection, Networking).
- **Types:** Know the different categories of system calls (Process control, File management, Device management, Information maintenance, Communication).
- **Examples:** Be familiar with examples of Windows and Unix system calls.

Study Tips:

- Understand the purpose of system calls as the gateway to OS services.
- Know the different categories of services accessed through system calls.

7. Operating System Design and Implementation

This section covers the process of creating an OS.

- **Problems:** Be aware that designing and implementing an OS is complex.
- **Design Goals:** Understand the different goals (User Goals vs. System Goals).
- **Mechanisms and Policies:** Understand the distinction between mechanisms (how to do something) and policies (what to do) and the importance of separating them.

- **Implementation Language:** Understand the advantages and disadvantages of implementing an OS using higher-level languages compared to assembly language.

Study Tips:

- Understand the challenges and goals involved in OS design.
- Know the trade-offs in choosing an implementation language.

8. System Boot

This section covers the process of starting a computer.

- **Concept:** Understand system boot as loading the kernel into memory and starting execution.
- **Process:** Understand the steps involved in the boot process (BIOS initialization, POST, finding/loading OS loader, OS initialization, loading device drivers, user access).
- **Benefits:** Understand the advantages of the system boot process.

Study Tips:

- Understand the sequence of events that occurs when a computer starts up.

9. Process

This section introduces the concept of a process as an executing program.

- **Definition:** Understand a process as a program in execution, the basic unit of work.
- **Structure:** Understand the structure of a process in memory (Stack, Heap, Data, Text sections) and what each section contains.
- **Program vs. Process:** Understand the difference between a program (code) and a process (dynamic instance of code execution).
- **Life Cycle:** Understand the different states a process goes through during its execution (Start, Ready, Running, Waiting, Terminated/Exit). Know the transitions between these states.
- **Process Control Block (PCB):** Understand the PCB as a data structure maintained by the OS for every process, containing information about the process state, ID, pointers, registers, scheduling info, memory info, I/O status, accounting info.

Study Tips:

- Understand the definition and components of a process.

- Master the different states in the process life cycle and the role of the PCB.

10. Process Scheduling

This section covers how the OS manages the execution of multiple processes.

- **Concept:** Understand process scheduling as the activity of selecting which process to run on the CPU. Know its importance in multiprogramming.
- **Process Scheduling Queues:** Understand the different queues where processes reside based on their state (Job queue, Ready queue, Device queues).
- **Two-State Process Model:** Understand the simplified model with Running and Not Running states.
- **Schedulers:** Understand the role of schedulers (system software handling scheduling). Know the three types: Long-Term Scheduler (Job Scheduler), Short-Term Scheduler (CPU Scheduler/Dispatcher), Medium-Term Scheduler (Process Swapping Scheduler). Understand their purpose and differences in speed and control over multiprogramming.
- **Context Switch:** Understand context switching as saving and restoring the state of a process to allow multiple processes to share the CPU. Know the information saved during a context switch.

Study Tips:

- Understand the purpose of process scheduling and the different queues involved.
- Know the roles of the different types of schedulers.
- Understand the concept and cost of context switching.

11. Process Operations

This section covers actions performed on processes.

- **Operations:** Understand different operations that can be performed on processes (Creation, Preemption, Blocking, Termination).
- **Process Creation:** Understand how processes are created (user request, system initialization, system call, batch job). Know the parent-child relationship and the use of fork().
- **Process Preemption:** Understand preemption (interrupting a running process to run another).

- **Process Blocking:** Understand blocking (process waiting for an event, e.g., I/O).
- **Process Termination:** Understand termination (process finishes or is stopped).

Study Tips:

- Understand the sequence of events in process creation, preemption, blocking, and termination.

12. Inter Process Communication (IPC)

This section covers how processes can communicate and synchronize.

- **Concept:** Understand IPC as a mechanism for processes to communicate and synchronize actions. Differentiate between Independent and Co-operating processes.
- **Methods:** Know the two main methods: Shared Memory and Message Passing.
- **Shared Memory:** Understand sharing a common memory space for communication. Be familiar with the Producer-Consumer problem as an example.
- **Message Passing:** Understand exchanging messages between processes without shared memory. Know basic primitives (send, receive). Understand communication links (Direct vs. Indirect via mailboxes/ports). Understand Synchronous (blocking) vs. Asynchronous (non-blocking) message passing.
- **Examples of IPC Systems:** Be aware of examples (Posix, Mach, Windows XP).
- **Client/Server Communication:** Understand client/server communication (clients request, servers respond). Know methods like Sockets, Remote Procedure Calls (RPCs), and Pipes.
- **Process Synchronization:** Understand process synchronization (handling concurrent access to shared data). Know the need for synchronized execution.
- **Critical Section Problem:** Understand the critical section (code accessing shared variables) and the problem (only one process in critical section at a time). Know the solution conditions (Mutual Exclusion, Progress, Bounded Waiting).
- **Synchronization Hardware:** Be aware of hardware support for critical sections.
- **Mutex Locks:** Understand Mutex Locks (software approach using acquire/release locks).

- **Classical Problems:** Be familiar with classical synchronization problems (Bounded Buffer, Dining Philosophers, Readers Writers).
- **Peterson's Solution:** Understand Peterson's solution for the two-process critical section problem (using turn and flag).
- **Semaphores:** Understand semaphores (integer variables with atomic wait and signal operations) for solving the critical section problem. Know Counting vs. Binary semaphores. Understand their advantages and disadvantages.

Study Tips:

- Understand the fundamental difference between shared memory and message passing IPC.
- Know the purpose of process synchronization and the critical section problem.
- Familiarize yourself with classical synchronization problems and basic solutions like Mutexes and Semaphores.

13. Threads

This section introduces threads as lightweight processes.

- **Concept:** Understand a thread as an execution unit within a process, sharing resources but having its own PC, stack, registers. Know they are Lightweight processes.
- **Types:** Know User Threads (managed by thread library in user space) and Kernel Threads (supported by the OS kernel).
- **Multithreading Models:** Understand how user threads are mapped to kernel threads (Many to One, One to One, Many to Many).
- **Thread Libraries:** Understand thread libraries (APIs for thread creation/management). Know implementation in user space vs. kernel space. Be aware of examples (POSIX Pthreads, Win32 threads, Java threads).
- **Benefits of Multithreading:** Know the advantages (Responsiveness, Resource sharing, Economy, Scalability, Smooth Context Switching).
- **Multithreading Issues:** Be aware of issues (Thread Cancellation, Signal Handling, fork() System Call, Security Issues).
- **Multicore Programming:** Understand multicore programming (creating concurrent systems for multicore processors). Know its advantages (performance, scalability).
- **Implicit Threading:** Understand implicit threading (compilers/libraries manage threads, hiding details). Know examples (OpenMP, Grand Central Dispatch (GCD)).

- **Language-based Threads:** Understand explicit threading support in languages (Java, Python, Go, Rust). Know concepts like Goroutines and Channels.
- **Threading Issues (Revisited):** Reiterate issues like complexity, concurrency complications, testing difficulties, unpredictable results, porting complications.

Study Tips:

- Understand the concept of threads and how they differ from processes.
- Know the different models for mapping user threads to kernel threads.
- Understand the benefits and challenges of using multithreading.

14. CPU Scheduling (Revisited)

This section provides a deeper dive into CPU scheduling algorithms and concepts.

- **Concept:** Reiterate CPU scheduling (selecting processes for CPU execution).
- **Dispatcher:** Understand the role of the dispatcher (giving CPU control to the selected process). Know Dispatch Latency.
- **Types of CPU Scheduling:** Understand Preemptive (can interrupt running process) vs. Non-Preemptive (process runs until it finishes or blocks). Know the circumstances under which scheduling decisions occur.
- **Scheduling Criteria:** Understand the different criteria for evaluating scheduling algorithms (CPU Utilization, Throughput, Turnaround Time, Waiting Time, Load Average, Response Time).
- **Scheduling Algorithms:** Master the common algorithms: First Come First Serve (FCFS), Shortest-Job-First (SJF - Non-Preemptive and Pre-emptive/Shortest Remaining Time First), Priority Scheduling (Preemptive and Non-Preemptive, Aging), Round Robin (RR), Multilevel Queue Scheduling, Multilevel Feedback Queue Scheduling. Understand how each algorithm works, its advantages, disadvantages, and potential problems (Convoy Effect, Starvation, Aging).
- **Thread Scheduling:** Understand scheduling of threads (user-level to kernel-level, and kernel-level by system scheduler). Know Lightweight Process (LWP). Understand Contention Scope (PCS vs. SCS) and Allocation Domain.
- **Multiple-Processor Scheduling:** Understand scheduling in systems with multiple CPUs. Know Homogeneous vs.

Heterogeneous processors. Understand Asymmetric vs. Symmetric Multiprocessing. Know Processor Affinity (Soft vs. Hard) and Load Balancing (Push vs. Pull Migration).

- **Multicore Processors:** Understand scheduling on multicore processors. Know Memory Stall. Understand Multithreaded processor cores (Coarse-Grained vs. Fine-Grained Multithreading).
- **Virtualization and Threading:** Understand scheduling in virtualized environments and its impact on performance.
- **Scheduling in Real Time Systems:** Understand scheduling in real-time systems (meeting deadlines). Know Hard vs. Soft real-time tasks. Understand different approaches (Static table-driven, Static priority-driven, Dynamic planning-based, Dynamic best effort).

Study Tips:

- Understand the goals of CPU scheduling and the different criteria for evaluation.
- Master the working principles of the major scheduling algorithms.
- Be able to compare and contrast different algorithms and their suitability for different scenarios.
- Understand the complexities of scheduling in multithreaded, multiprocessor, and real-time systems.

15. Deadlock

This section covers the situation where processes are blocked indefinitely.

- **Concept:** Understand deadlock (set of blocked processes waiting for resources held by others).
- **Characterization (Coffman Conditions):** Understand the four necessary and sufficient conditions for deadlock (Mutual Exclusion, Hold and Wait, No Preemption, Circular Wait). Understand how each condition contributes to deadlock.
- **Prevention:** Understand preventing deadlock by eliminating one of the Coffman conditions.
- **Avoidance:** Understand avoiding deadlock (e.g., Banker's Algorithm) by checking for safe states before granting resources.
- **Detection:** Understand detecting deadlock (checking for cycles in Resource Allocation Graph).
- **Recovery:** Understand recovering from deadlock (Preemption, Rollback, Killing processes).

2025 Latest Edition

e-Booklet

Subect – Commerce

Professors Adda

Updated as per NEW UGC trend

Highly Useful for UGC NET JRF / SET / PG (CUET (PG) Asst Professor

- **Livelock:** Understand livelock as a variant of deadlock where processes change state without making progress.

Study Tips:

- Understand the four conditions for deadlock thoroughly.
- Know the different strategies for preventing, avoiding, detecting, and recovering from deadlock.

16. Memory Management (Revisited)

This section provides a deeper dive into how the OS manages memory.

- **Concept:** Reiterate memory management (handling primary memory, moving processes).
- **Process Address Space:** Understand process address space (logical addresses). Know Symbolic, Relative, and Physical addresses. Understand the role of the MMU in translating virtual to physical addresses.
- **Static vs Dynamic Loading/Linking:** Reiterate these concepts in the context of memory allocation.
- **Swapping:** Understand swapping (moving processes between main memory and disk) and its purpose (memory compaction, running multiple processes).
- **Memory Allocation:** Understand how main memory is partitioned and allocated to processes (Single-partition, Multiple-partition).
- **Fragmentation:** Understand fragmentation (free memory broken into small pieces) and its types (External, Internal).
- **Paging:** Understand paging (non-contiguous allocation with fixed-size pages/frames). Know Address Translation (logical to physical). Understand its advantages and disadvantages (reduces external fragmentation, internal fragmentation, extra space for page table).
- **Segmentation:** Understand segmentation (dividing jobs into variable-sized segments). Know its similarity and differences from paging (variable size, different logical address space, suffers from external fragmentation).
- **Segmentation with Paging:** Understand combining segmentation and paging.
- **Contiguous Memory Allocation:** Understand contiguous allocation (all memory together). Know Fixed-sized and Variable-sized partitions and their pros/cons.
- **Demand Paging:** Understand demand paging (loading pages only when needed) and its purpose (implementing virtual memory). Know Page Faults and Thrashing.

- **Methods:** Be aware of general security measures (patch updates, antivirus, firewall, secure accounts).
- **Protection:** Understand protection (controlling access to resources). Know the need and role of protection. Understand the distinction between mechanism and policy in protection.
- **Access Matrix:** Understand the access matrix model for defining access rights (domains, objects, access rights). Know how it implements policies and controls domain switching.
- **Access Control:** Understand access control (guaranteeing user identity and appropriate access). Know its components (authentication and authorization). Be aware of the risks of weak access control (access mining). Know key considerations for access control policy (dynamic adaptation, consistency across environments, security analytics).
- **Types of Access Control:** Know different models (Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC), Attribute Based Access Control (ABAC)). Understand the principle of each model.
- **Access Control Solutions:** Be aware of technologies supporting access control models.
- **Why Authorization Remains a Challenge:** Understand the difficulties in implementing and maintaining authorization.
- **Revocation of Access Rights:** Understand revoking access rights. Know questions about revocation (immediate vs. delayed, selective vs. general, partial vs. total, temporary vs. permanent). Understand schemes for implementing revocation (access lists, capabilities with reacquisition/back-pointers/indirection/keys).
- **Threats:** Understand threats (program threats, system threats). Know types of Program Threats (Virus, Trojan Horse, Trap Door, Logic Bomb) and System Threats (Worm, Port Scanning, Denial of Service).
- **Security Measures Taken:** Understand measures taken at different levels (Physical, Human, Operating system, Networking System). Know Anti Malware programs and Firewalls.
- **Network Security Threats:** Understand network-delivered threats (Passive vs. Active). Know how attackers gain access (email threats, exploiting vulnerabilities). Understand Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks.
- **Cryptography:** Understand cryptography (encrypting/decrypting data). Know plaintext, encryption, cipher text, decryption, cryptanalysis. Understand how encryption works (algorithm + key).

Know cryptography techniques (Symmetric Encryption, Asymmetric Encryption, Hashing).

- **Authentication (Revisited):** Reiterate authentication (identifying users). Know methods (Username/Password, User card/key, User attribute). Understand One Time passwords.
- **Program Threats (Revisited):** Reiterate program threats (malicious programs).
- **Computer Security Classifications:** Understand security classifications (e.g., U.S. DoD levels A, B, C, D). Know the principle behind each classification level.

Study Tips:

- Understand the fundamental concepts of OS security, including authentication, authorization, and protection.
- Know the different types of threats and vulnerabilities.
- Familiarize yourself with basic cryptography concepts.
- Understand different access control models.

28. Virtual Machine

This section covers virtual machines.

- **Concept:** Understand a VM (emulated computer system using software). Know host vs. guest machines.
- **Uses:** Understand what VMs are used for (running different OS, testing, server virtualization, risky tasks).
- **How They Work:** Understand how VMs run as processes on a physical machine's OS, managed by a hypervisor.
- **Advantages and Disadvantages:** Know the pros and cons of using VMs (multiple OS, resource consolidation, legacy apps, disaster recovery, performance issues, infrastructure requirements).
- **Types:** Know Process Virtual Machines (running a single process) and System Virtual Machines (fully virtualized physical machine).
- **Types of Virtualization:** Understand different types of virtualization (Hardware, Software, Storage, Network, Desktop).
- **Container vs Virtual Machine:** This is a key comparison. Understand containers (packaging app + dependencies, sharing OS kernel) and how they differ from VMs (overhead, boot speed, size, isolation). Know use cases for each.
- **Virtualization (Revisited):** Understand OS virtualization (running OS from the network using a virtual disk). Know the components (server, client, virtual disk, supporting components). Understand how OS virtualization works (connecting to server, connecting virtual disk, streaming OS content).

Study Tips:

- Understand the concept of virtualization and virtual machines.
- Know the difference between process and system VMs.
- Master the comparison between containers and virtual machines.

29. Linux Operating Systems

This section focuses on the Linux OS.

- **Concept:** Understand Linux as a version of UNIX, open source.
- **Components:** Know the primary components (Kernel, System Library, System Utility).
- **Kernel Mode vs. User Mode:** Understand these two modes of execution and the privileges in each.
- **Basic Features:** Know the important features of Linux (Portable, Open Source, Multi-User, Multiprogramming, Hierarchical File System, Shell, Security).
- **Architecture:** Understand the layered architecture of a Linux system (Hardware layer, Kernel, Shell, Utilities).
- **Design Principles:** Understand the design principles (UNIX compatibility, fast and efficient, standardization - POSIX).
- **System Components:** Understand the three code parts (Kernel, System library, System Utility) and their roles.
- **Linux Process Management:** Understand process management in Linux (monitoring, managing, maintaining processes). Know Process ID (PID). Understand Multitasking (preemptive, scheduler, time slice). Know Types of Processes (Interactive - foreground/background, System process/Daemon).
- **Linux Scheduling:** Understand the Linux scheduler (priority based). Know static and dynamic priorities, goodness. Understand the evolution of the Linux scheduler (O(n) in 2.4, O(1) in 2.6, Completely Fair Scheduler (CFS)). Know scheduler policies (SCHED_FIFO, SCHED_RR, SCHED_OTHER/NORMAL, SCHED_BATCH, SCHED_IDLE). Understand the O(1) algorithm and its features. Understand CFS (fairness, virtual runtime, red-black tree, sleeper fairness). Know scheduler classes (CFS, RT). Understand Load balancer. Understand Interactivity (estimator, bonus/penalty, re-inserting interactive tasks).
- **Memory Management:** Understand memory management in Linux (virtual memory, demand paging, allocation, file mapping). Know the nommu case.
- **Linux File System:** Understand the Linux file system (structured collection of files). Know the root directory. Understand its

structure (hierarchical). Know Namespace. Understand its features (paths, partitions/drives, case sensitivity, file extensions, hidden files). Know Types of Linux File System (Ext, Ext2, Ext3, Ext4, JFS, ReiserFS, XFS, btrfs, swap).

- **Linux-Input & Output:** Understand I/O in Linux (devices appear as files). Know device classes (block, character, network). Understand the structure of the device-driver system. Understand Block Devices (random access, fixed blocks, scheduling - elevator, deadline). Understand Character Devices (serial access, tty_struct, line discipline). Understand Network Devices (networking subsystem).
- **Methods in Interprocess Communication:** Understand IPC methods in Linux (Pipes, Names Pipes, Message Queuing, Semaphores, Shared memory, Sockets).
- **Network Structure:** Understand the network structure in Linux (networking subsystem, network virtualization - OpenStack Neutron, SDN, white box switches).
- **Windows Operating Systems (Brief Comparison):** Be aware of Windows as a graphical OS and its history/versions.

Study Tips:

- Understand the core components and architecture of the Linux OS.
- Focus on process management and scheduling in Linux, including the evolution of the scheduler.
- Understand the Linux file system structure and types.
- Familiarize yourself with I/O and IPC in the Linux environment.

30. Windows Operating Systems

This section focuses on the Windows OS.

- **Concept:** Understand Windows as a graphical OS by Microsoft.
- **What is an Operating System (Revisited):** Reiterate the OS definition.
- **History:** Know the history of Windows versions (early versions, 3.x, 9x, NT, XP, Vista, 7, 8/8.1, 10).
- **Design Principles:** Understand the design goals (security, reliability, application compatibility - Windows/POSIX, high performance, extensibility, portability, international support).
- **Main Components:** Know the main components of Windows.
- **Fast User Switching:** Understand Fast User Switching (switching users without logging out).
- **Terminal Services:** Understand Terminal Services (accessing remote applications/desktops). Know its purpose (centralized

management, remote access). Understand its history and limitations. Know client applications (Remote Assistance, Remote Desktop). Understand Terminal Services on client vs. server versions. Know its disadvantages (server power, network, reliability, performance). Understand Microsoft Terminal Services features (Group Policy, Remote Administration, Remote Desktop Protocol (RDP), Session Directory component).

- **Windows File System:** Understand the Windows file system (controlling data storage/retrieval). Know its types (FAT12, FAT16, FAT32, NTFS, exFAT). Understand their characteristics and comparisons (compatibility, ideal use, security).

Study Tips:

- Understand the history and evolution of Windows.
- Know the key design principles and features of Windows.
- Familiarize yourself with Terminal Services and its applications.
- Understand the different types of Windows file systems.

31. Distributed Systems

This section covers systems with multiple interconnected nodes.

- **Concept:** Understand distributed systems (multiple physically separate nodes linked by a network).
- **Types:** Know Client/Server Systems and Peer to Peer Systems.
- **Advantages and Disadvantages:** Know the pros and cons of distributed systems (resource sharing, scalability, reliability vs. security, data loss, complexity, overloading).
- **NETWORK BASED OPERATING SYSTEM:** Understand Network Operating Systems (NOS) coordinating multiple computers across a network. Know the two concepts of NOS (specialized OS for network device vs. OS for sharing resources).
- **NETWORK DEVICE OPERATING SYSTEM:** Be aware of examples of OS for network devices.
- **Historical Network Operating System:** Understand historical NOS (e.g., client-server, peer-to-peer).
- **Why Build a Distributed System:** Understand the motivations for building distributed systems (combining computing power, resource sharing, performance, reliability, expandability).
- **System Models:** Be aware of different system models (minicomputer, workstation, processor pool).
- **Where Knowledge is Useful:** Know areas where knowledge of distributed OS is applicable.

- **Lack of Global Knowledge:** Understand the challenges due to communication delays and lack of global clock/state.
- **Naming:** Understand naming in distributed systems (named objects, namespace, mapping).
- **Scalability:** Understand scalability (system efficiency with increasing users/resources).
- **Compatibility:** Understand different levels of compatibility (Binary, Execution, Protocol).
- **Process Synchronization:** Understand the need for synchronization mechanisms in distributed systems.
- **Distributed Resource Management:** Understand managing resources across distributed nodes (Data migration, Computation migration).
- **Security:** Understand security in distributed systems (Authentication, Authorization).
- **Structuring:** Understand different structuring approaches (monolithic kernel, collective kernel, object oriented, client-server).
- **Communication Networks:** Understand communication networks (WAN, LAN) and traditional OS implementation of TCP/IP.
- **Communication Models:** Understand communication models (message passing, RPC).
- **Message Passing Primitives:** Understand message passing primitives (Send, Receive). Know characteristics (buffered/unbuffered, blocking/nonblocking, reliable/unreliable, synchronous/asynchronous). Be familiar with Unix socket I/O primitives.
- **RPC (Remote Procedure Call):** Understand RPC (virtual procedure call model). Know its purpose (hiding communication details). Understand RPC Issues (Stubs, Binding method, Parameter/result passing, Error handling).
- **Communication Protocols:** Understand communication protocols (layered approach, e.g., ISO protocol stack - Physical, Data-Link, Network, Transport, Session, Presentation, Application layers).
- **Design Issues:** Understand design issues of distributed systems (Heterogeneity, Openness, Scalability, Security, Failure Handling, Concurrency, Transparency).
- **Distributed File System (DFS) (Revisited):** Reiterate DFS (data distributed across servers, accessed transparently).

Study Tips:

- Understand the motivations and challenges of distributed systems.
- Know different types of distributed systems and their architectures.

AMRIUT BOOKLET

PROFESSORS ADDA

What is this, why read it?

- AMRIT Booklet is designed on PYQ pattern by extracting exam-useful essence from all major books of the subject at one place. You don't have to read books now.
- This is not just an ordinary booklet but a top-level rstudy tool, specially designed for those students who want quick revision, exam-time recall and concept clarity.
- In this, you will get the "amrit nichuran" of every important topic – that is, the same things which are asked again and again in the exam.
- This booklet brings together Core Concepts, Keywords, Thinkers, Definitions and Chronology of every subject at one place – and that too in a very crisp question and answer style.

Benefits & Features:

- ✓ Super Quick Revision Tool
- ✓ Exam Time Confidence Booster
- ✓ High Retention Format –
- ✓ 100% Exam-Oriented – No Extra, No Fluff

ALL INDIA RANK

How to make best use?

✅ First read the Amrit page of the topic from the guide
✅ Memorize the Keywords along with the Concepts
✅ Solve MCQs from that topic on the same day
✅ Revise only from this before the exam — Time Saving, Score Boosting

Bonus Insides

Who is this for?

- ✅ NET / SET/ PGT
- ✅ Assistant Professor Candidates
- ✅ Those who have less time but want strong results and the syllabus to be completed

This booklet is for all those who do not just want to read, but want to "read right".

What will you get in it?

- One Page One Topic Format – One complete topic clear on each page
- Updated as per latest changes of 2025

PROFESSORS ADDA

Available in Digital PDF + Print Format

Book Now | DM | WhatsApp | Download from the link

sample Notes/
Expert Guidance/Courier Facility Available

Download PROFESSORS ADDA APP

+91 7690022111 +91 9216228788

Discrete Structures and Optimization One Liner

Mathematical Logic: Propositional and Predicate Logic

Q.1 What is a proposition?

A. A declarative statement; either true/false.

Q.2 What are logical operators?

A. Symbols connecting propositions (e.g., ∧ ,∨ ,¬).

Q.3 What is a 'truth table'?

A. Table showing all possible truth values.

Q.4 What is Negation (¬)?

A. Reverses the truth value.

Q.5 What is Conjunction (∧)?

A. 'AND'; true if both are true.

Q.6 What is Disjunction (∨)?

A. 'OR'; true if one is true.

Q.7 What is Exclusive OR (XOR)?

A. True if one is true, not both.

Q.8 What is Implication (→)?

A. 'If-then'; false only if T → F.

Q.9 In P → Q, P is the _____.

A. Antecedent or premise.

Q.10 In P → Q, Q is the _____.

A. Consequent or conclusion.

Q.11 What is Biconditional (↔)?

A. 'If and only if' (iff).

Q.12 When is P ↔ Q true?

A. When P and Q are same.

Q.13 What is a 'tautology'?

A. A proposition that is always true.

Q.14 What is a 'contradiction'?

A. A proposition that is always false.

Q.15 What is a 'contingency'?

A. A proposition that is neither tautology nor contradiction.

Q.16 What is 'Predicate Logic'?

A. Logic involving predicates and quantifiers.

Q.17 What is a 'predicate'?

A. A property of the subject.

Q.18 Predicate logic deals with _____ of statements.

A. Quantifiers and predicates.

Q.19 What is the 'universe of discourse'?

A. The domain of variables.

Q.20 What does predicate P(x) mean?

A. P is a property of x.

Mathematical Logic: Propositional Equivalences, Normal Forms

Q.21 What are 'propositional equivalences'?

A. Two statements with same truth values.

Q.22 What is the symbol for logical equivalence?

A. ≡ or ⇔

Q.23 What are De Morgan's Laws?

A. Rules for negating conjunctions/disjunctions.

Q.24 ¬(P∧ Q) is equivalent to?

A. ¬P∨ ¬Q.

Q.25 ¬(P∨ Q) is equivalent to?

A. ¬P∧ ¬Q.

Q.26 What is the 'commutative law'?

A. P ∨ Q ≡ Q ∨ P.

Q.27 What is the 'associative law'?

A. (P ∨ Q) ∨ R ≡ P ∨ (Q ∨ R).

Q.28 What is the 'distributive law'?

A. P ∧ (Q ∨ R) ≡ (P ∧ Q) ∨ (P ∧ R).

Q.29 What is the 'contrapositive' of P → Q?

A. $\neg$Q → $\neg$P.

Q.30 What is the 'converse' of P → Q?

A. Q → P.

Q.31 What is the 'inverse' of P → Q?

A. $\neg$P → $\neg$Q.

Q.32 A proposition is logically equivalent to its _____.

A. Contrapositive.

Q.33 What is a 'normal form'?

A. A standard way of writing formulas.

Q.34 What is 'Disjunctive Normal Form' (DNF)?

A. Disjunction of conjunctions of literals.

Q.35 What is 'Conjunctive Normal Form' (CNF)?

A. Conjunction of disjunctions of literals.

Q.36 What is a 'literal'?

A. A variable or its negation.

Q.37 Every propositional formula has an equivalent _____.

A. DNF and CNF.

Q.38 What is 'Principal Disjunctive Normal Form' (PDNF)?

A. DNF with all variables in each term.

Q.39 What is 'Principal Conjunctive Normal Form' (PCNF)?

A. CNF with all variables in each clause.

Q.40 What is 'functional completeness'?

A. Set of operators can express all propositions.

Q.41 Is {∧ ,∨ ,¬} functionally complete?

A. Yes.

Q.42 Is {∧ ,¬} functionally complete?

A. Yes.

Q.43 Is {∨ ,¬} functionally complete?

A. Yes.

Q.44 What is NAND operator (↑)?

A. Negation of AND (¬(P∧ Q)).

Q.45 What is NOR operator (↓)?

A. Negation of OR (¬(P∨ Q)).

Mathematical Logic: Predicates and Quantifiers, Nested Quantifiers, Rules of Inference

Q.46 What is a 'quantifier'?

A. Specifies quantity of elements in domain.

Q.47 What are the two main types of quantifiers?

A. Universal and Existential.

Q.48 What is the 'Universal Quantifier' (∀)?

A. "For all" or "for every".

Q.49 What is the 'Existential Quantifier' (∃)?

A. "There exists" or "for some".

Q.50 What is a 'bound variable'?

A. A variable within a quantifier's scope.

Q.51 What is a 'free variable'?

A. A variable not bound by a quantifier.

Q.52 How do you negate ∀xP(x)?

A. ∃x¬P(x).

Q.53 How do you negate ∃xP(x)?

A. ∀x¬P(x).

Q.54 What are 'nested quantifiers'?

A. Quantifiers within the scope of another.

Q.55 Does the order of nested quantifiers matter if they are

different?

A. Yes, ∀ x∃ y is different from ∃ y∀ x.

Q.56 Does the order of nested quantifiers matter if they are the same?

A. No.

Q.57 What are 'rules of inference'?

A. Rules for deriving valid conclusions.

Q.58 What is 'modus ponens'?

A. If P and P$\to$Q, then Q.

Q.59 What is 'modus tollens'?

A. If $\neg$Q and P$\to$Q, then $\neg$P.

Q.60 What is 'hypothetical syllogism'?

A. If P$\to$Q and Q$\to$R, then P$\to$R.

Q.61 What is 'disjunctive syllogism'?

A. If P$\lor$Q and $\neg$P, then Q.

Q.62 What is 'addition' rule of inference?

A. If P, then P$\lor$Q.

Q.63 What is 'simplification' rule of inference?

A. If P$\land$Q, then P.

Q.64 What is 'universal instantiation'?

A. From ∀ xP(x), infer P(c).

Q.65 What is 'universal generalization'?

A. From P(c) for arbitrary c, infer ∀ xP(x).

Q.66 What is 'existential instantiation'?

A. From ∃ xP(x), infer P(c) for some c.

Q.67 What is 'existential generalization'?

A. From P(c), infer ∃ xP(x).

Q.68 What is a 'valid argument'?

A. Conclusion follows logically from premises.

Q.69 An argument is valid if premises are true and conclusion is _____.

A. True.

Q.70 What is a 'fallacy'?

A. An error in reasoning.

Sets and Relations: Set Operations

Q.71 What is a 'set'?

A. A well-defined collection of distinct objects.

Q.72 What are 'elements' or 'members'?

A. The objects in a set.

Q.73 What is a 'roster method' of set representation?

A. Listing all elements (e.g., {1, 2, 3}).

Q.74 What is a 'set-builder notation'?

A. Describing elements by a property.

Q.75 What is an 'empty set' (∅ or {})?

A. A set with no elements.

Q.76 What is a 'subset' ($A \subseteq B$)?

A. All elements of A are in B.

Q.77 What is a 'proper subset' ($A \subset B$)?

A. A is a subset of B, but A □= B.

Q.78 What is a 'universal set' (U)?

A. Set containing all possible elements.

Q.79 What is 'set union' (A ∪ B)?

A. All elements in A or B.

Q.80 What is 'set intersection' (A ∩ B)?

A. All elements in both A and B.

Q.81 What is 'set difference' (A - B)?

A. Elements in A but not in B.

Q.82 What is 'set complement' (A')?

A. Elements in U but not in A.

Q.83 What are 'disjoint sets'?

A. Sets with no common elements.

Q.84 What is the 'cardinality' of a set?

A. The number of elements in it.

Q.85 What is a 'power set' P(A)?

A. The set of all subsets of A.

Q.86 If |A|=n, what is |P(A)|?

A. 2^n.

Q.87 What is 'Cartesian product' (A x B)?

A. Set of all ordered pairs (a, b).

Q.88 If |A|=m and |B|=n, what is |A x B|?

A. m * n.

Q.89 What is a 'Venn diagram'?

A. A diagram representing sets as circles.

Q.90 What do 'set identities' show?

A. Relationships between set operations.

Q.91 What is the 'commutative law' for sets?

A. $A \cup B = B \cup A$.

Q.92 What is the 'associative law' for sets?

A. $(A \cup B) \cup C = A \cup (B \cup C)$.

Q.93 What is the 'distributive law' for sets?

A. $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

Q.94 What are 'De Morgan's laws' for sets?

A. $(A \cup B)' = A' \cap B'$.

Q.95 What is a 'multiset'?

A. A set allowing duplicate elements.

Sets and Relations: Representation and Properties of Relations

Q.96 What is a 'binary relation' from A to B?

A. A subset of the Cartesian product A x B.

Q.97 A relation on a set A is a subset of _____.

A. A x A.

Q.98 How can a relation be represented?

A. Roster method, matrix, directed graph.

Q.99 What is a 'zero-one matrix' representation of a relation?

A. $M_{ij} = 1$ if $(a_i, b_j) \in R$.

Q.100 What is a 'directed graph' (digraph)?

A. A set of vertices and directed edges.

Professors Adda

PAID STUDENTS' BENEFITS

- Access to PYQs of the last 1 year

- Entry into Quiz Group + Premium Materials
- 20% Discount on Future Purchases / For Referring a Friend

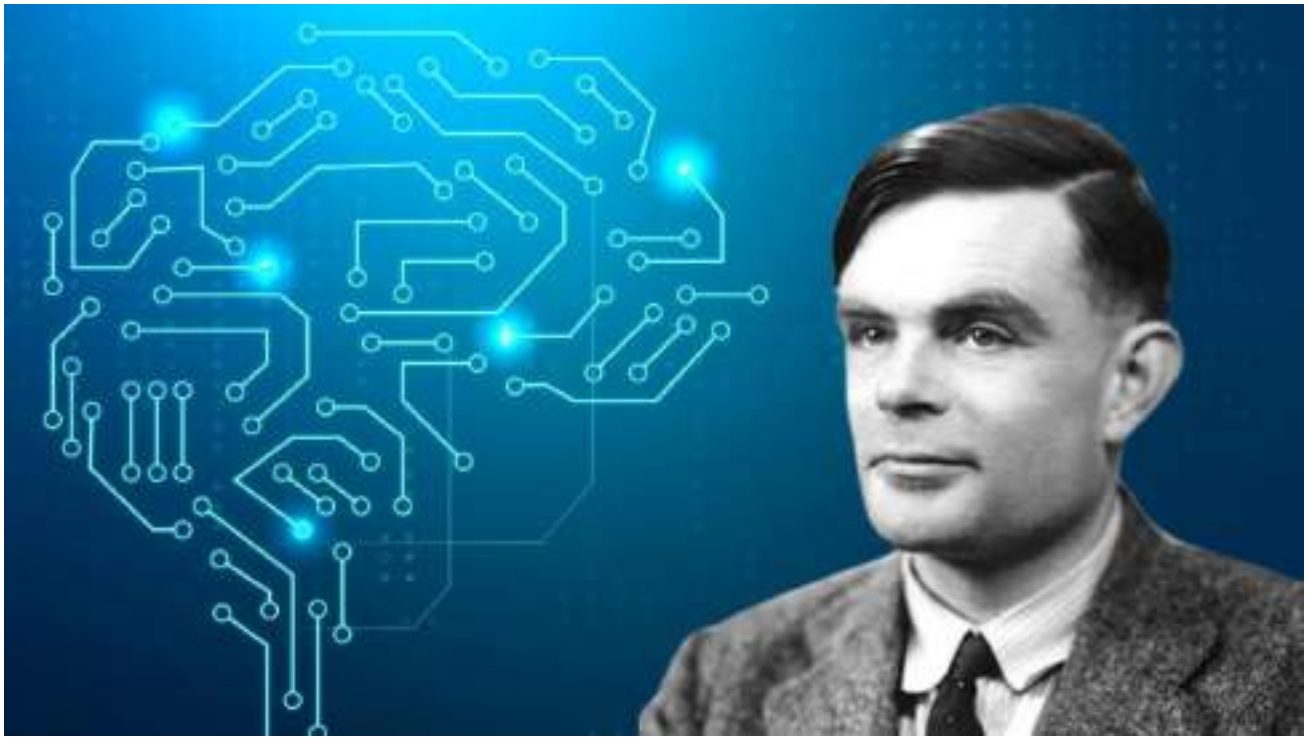
- Access to Current Affairs + Premium Study Group

NOTE: Please share your *Fee Receipt* or Payment Screenshot for activation.

Call/Whapp 76900-22111, 9216228788

COMPUTER SCIENCE AND APPLICATIONS TOPER TOOL KIT SUPER REVISION

Thinker 1: Alan Turing (1912–1954)



1. **Introduction**: Alan Turing was a British mathematician, logician, and computer scientist, often hailed as the father of theoretical computer science and artificial intelligence. His work laid the theoretical groundwork for modern computers and provided a formal concept of algorithm and computation.
2. **Major Contributions:**

- **Turing Machine**: Introduced a theoretical model of computation that could simulate any algorithmic process. This abstract machine became the foundation for the theory of computation. [1]
- **Codebreaking at Bletchley Park**: Played a crucial role in breaking German ciphers during World War II, most notably the Enigma code. His work significantly shortened the war and saved millions of lives.
- **Turing Test**: Proposed a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. [2]

3. **Key Concepts:**
   - **Computability**: Turing's work addressed the question of which problems are solvable by algorithms. The Halting Problem, which he proved to be unsolvable, is a cornerstone of this field. [3]
   - **Turing Machine**: A mathematical model of a hypothetical device that manipulates symbols on a strip of tape according to a table of rules. It is the basis for the Church-Turing thesis. [4]
   - **Artificial Intelligence**: His 1950 paper "Computing Machinery and Intelligence" introduced the concept of the Turing Test, fundamentally shaping the philosophy and goals of AI. [5]

4. **Key Books/Works:**
   - *On Computable Numbers, with an Application to the Entscheidungsproblem* (1936)
   - *Computing Machinery and Intelligence* (1950)

Thinker 2: John von Neumann (1903–1957)

1. **Introduction**: John von Neumann was a Hungarian-American mathematician, physicist, and computer scientist. His broad contributions influenced numerous fields, but he is most famous in computer science for the von Neumann architecture, which remains the basis for nearly all

modern computers.

2. **Major Contributions:**
   - **Von Neumann Architecture**: Described a computer architecture design featuring a processing unit (CPU), a control unit, memory, storage, and input/output. The key concept was the stored-program computer, where both data and instructions are stored in the same memory.
   - **Game Theory**: Co-authored the foundational text on game theory, which has applications in economics, AI, and military strategy.
   - **Merge Sort Algorithm**: Developed the efficient, divide-and-conquer merge sort algorithm in 1945.
3. **Key Concepts:**
   - **Stored-Program Concept**: The fundamental principle that instructions and data should be stored together in a computer's memory, allowing for program modification and general-purpose computing. [7]
   - **Cellular Automata**: Investigated self-replicating systems using cellular automata, which are discrete models studied in computability theory and complexity theory.
   - **Monte Carlo Method**: Developed and popularized methods for using random number generation to solve deterministic problems, a technique vital in scientific computing.

4. **Key Books/Works:**
   - *Theory of Games and Economic Behavior* (with Oskar Morgenstern, 1944)
   - *First Draft of a Report on the EDVAC* (1945)
   - *The Computer and the Brain* (published posthumously in 1958)

Thinker 3: Edsger W. Dijkstra (1930–2002)

1. **Introduction**: Edsger Dijkstra was a Dutch computer scientist and a pioneer of modern computing. He is recognized for his contributions to

algorithm design, programming languages, operating systems, and for his advocacy of structured programming.

2. **Major Contributions:**
   - **Dijkstra's Algorithm**: Created an algorithm to find the shortest paths between nodes in a weighted graph, a fundamental tool in network routing and logistics. [8]
   - **Semaphores**: Introduced the semaphore concept as a solution to the critical-section problem in concurrent programming, preventing race conditions. [9]
   - **Structured Programming**: Argued for the elimination of the GOTO statement in his influential letter "Go To Statement Considered Harmful," promoting a more disciplined and structured programming style.
3. **Key Concepts:**
   - **Shortest Path Algorithm**: An algorithm that finds the path with the lowest cumulative cost (e.g., distance, time) between two nodes in a graph. [10]
   - **Deadlock Avoidance**: Developed the Banker's Algorithm, a resource allocation and deadlock avoidance algorithm used in operating systems. [11]
   - **Concurrency and Synchronization**: His work on mutual exclusion, semaphores, and monitors laid

the foundation for modern operating systems and concurrent processing. [12]

4. **Key Books/Works:**
   - *A Discipline of Programming* (1976)
   - *Selected Writings on Computing: A Personal Perspective* (1982)

Thinker 4: Donald Knuth (1938–Present)

1. **Introduction**: Donald Knuth is an American computer scientist and mathematician, widely regarded as the "father" of the analysis of algorithms. His multi-volume work, *The Art of Computer Programming*, is considered the definitive bible on the subject.

2. **Major Contributions:**
   - ***The Art of Computer Programming* (TAOCP)**: Authored this seminal series of books that provides a comprehensive and rigorous mathematical analysis of computer algorithms.
   - **TeX and METAFONT**: Created the TeX typesetting system and the METAFONT font-design system to produce high-quality scientific and mathematical documents.
   - **Asymptotic Notation**: Popularized and formalized the use of Big-O, Big-Omega, and Big-Theta notations for analyzing the complexity and performance of algorithms. [13]
3. **Key Concepts:**
   - **Analysis of Algorithms**: Knuth pioneered the rigorous mathematical analysis of algorithms, focusing on their computational complexity (time and space). [14]
   - **Literate Programming**: A programming paradigm he introduced that treats a program as a piece of literature, understandable by human beings, where code and documentation are interwoven.
   - **Knuth-Morris-Pratt (KMP) Algorithm**: Co-developed an efficient string-searching algorithm that searches for occurrences of a "word" within a main "text." [15]

4. **Key Books/Works:**
   - *The Art of Computer Programming, Vol. 1: Fundamental Algorithms* (1968)
   - *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms* (1969)
   - *The Art of Computer Programming, Vol. 3: Sorting and Searching* (1973) [16]
   - *Concrete Mathematics: A Foundation for Computer Science* (with Oren Patashnik and Ronald Graham, 1989)

Thinker 5: Grace Hopper (1906–1992)

1. **Introduction**: Grace Hopper was an American computer scientist and a United States Navy rear admiral. A pioneer of computer programming, she was one of the first programmers of the Harvard Mark I computer and developed the first compiler for a computer programming language.
2. **Major Contributions:**
   - **First Compiler**: Created the A-0 System, the first compiler, which translated mathematical code into machine-readable code. This paved the way for modern programming languages. [17]
   - **COBOL (Common Business-Oriented Language)**: Her work on FLOW-MATIC, an early English-like data processing language, directly influenced the design of COBOL, which became a ubiquitous language for business applications.
   - **Debugging**: Popularized the term "debugging" after a moth was removed from a relay in the Mark II computer.
3. **Key Concepts:**
   - **Machine-Independent Programming Languages**: Championed the idea of programming languages that were closer to human language rather than machine code, making programming more accessible.
   - **Compiler**: A program that translates source code written in a high-level language into a lower-level

language (like machine code) to create an executable program. [18]

- **Validation**: Developed standardized testing and validation procedures for programming languages to ensure they conformed to specifications, a precursor to modern software testing. [19]

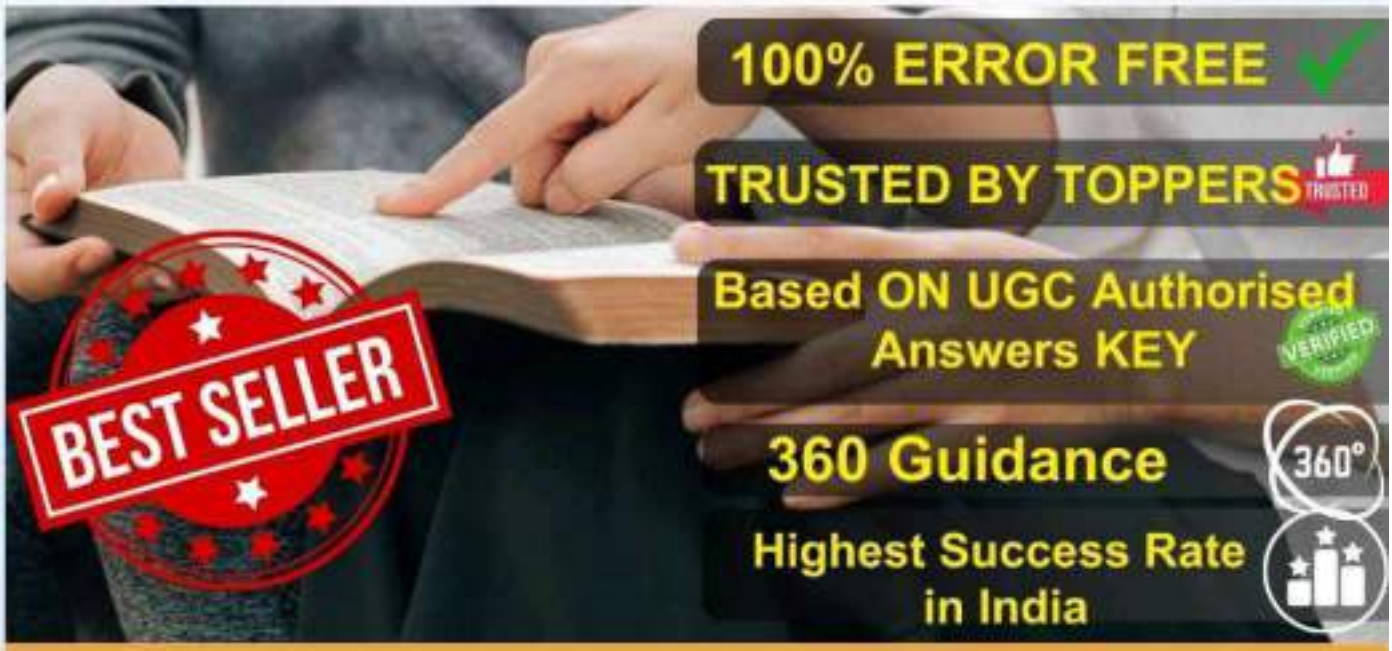
4. **Key Books/Works:**
   - Hopper did not author major books but published numerous papers and lectures. Her ideas are primarily known through her influential work on language standards committees and her extensive public speaking.

10 YEAR'S PYQ

ALL SUBJECT AND PAPER I

UGC NET

BOTH MEDIUM AVAILABLE



UGC NET COMPUTER SCIENCE

UGC NET Computer Science: Questions 51-65 with Explanations

Q.51 Consider the following statements regarding Agent systems
A. Agent system comprises of an agent and an environment on which it acts
B. The controller part of an agent receives percepts from its body and sends commands to the environment
C. Agents act in the world through actuators which are non-noisy and always reliable.
D. The actuators of an agent convert stimuli into percepts
Choose the correct answer from the options given below:
(1) A, B Only
(2) B, D Only
(3) C, D Only
(4) B, C, D Only

Correct Answer: (1)

Explanation:

- **Concept:** This question tests the fundamental concepts of an intelligent agent in Artificial Intelligence. An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.

- **Fact (A):** Statement A is correct. The basic model of an agent system consists of an **agent** and the **environment** it interacts with.
- **Fact (B & D):** The agent's "body" has **sensors** to perceive stimuli from the environment, which are converted into **percepts**. The agent's "brain" or **controller** processes these percepts and sends commands to its **actuators** to perform actions. Statement B is incorrect because the controller sends commands to the actuators, not directly to the environment. Statement D is incorrect because **sensors** convert stimuli into percepts, not actuators.
- **Conclusion:** Statement C is incorrect as a general rule; actuators in the real world can be noisy and are not always reliable. Therefore, only statements A and B (with the understanding that commands are sent to actuators) are the most accurate descriptions provided.

Q.52 Which of the following trees are height balanced?
A. Binary Search Tree
B. AVL Tree
C. Red-Black Tree
D. B Tree
Choose the correct answer from the options given below:
(1) A and D Only
(2) A, B and D Only
(3) C and D Only
(4) B and C Only

Correct Answer: (4)

Explanation:

- **Concept:** A height-balanced binary tree is a binary tree in which the heights of the left and right subtrees of any node differ by at most one. This property ensures that the tree's height remains logarithmic in the number of nodes, guaranteeing efficient search, insertion, and deletion operations.
- **AVL Tree (B):** By definition, an AVL tree is a self-balancing binary search tree where the difference between the heights of the left and right subtrees (the balance factor) for any node is -1, 0, or +1.
- **Red-Black Tree (C):** A Red-Black Tree is another type of self-balancing binary search tree that uses node coloring (red or black) to enforce properties that guarantee the longest path from the root to any leaf is no more than twice as long as the shortest path, thus keeping it approximately balanced.
- **Conclusion:** A standard Binary Search Tree (A) provides no guarantee of being balanced. A B-Tree (D) is a multi-way search tree that is always balanced, but it is not a binary tree. The question specifically asks which of the given trees are height-balanced, and AVL and Red-Black trees are the canonical examples of height-balanced binary trees.

Q.53 The longest common subsequence of (1,2,3,2,4,1,2) and (2,4,3,1,2,1) is
(1) 2,1,2,3
(2) 1,3,2,1

(3) 2,3,2,1
(4) 2,3,1,2,1

Correct Answer: (3)

Explanation:

- **Concept:** The Longest Common Subsequence (LCS) problem is a classic computer science problem to find the longest subsequence common to all sequences in a set of sequences. A subsequence is derived from another sequence by deleting zero or more elements without changing the order of the remaining elements.
- **Theory:** This problem is typically solved using dynamic programming. A 2D table is constructed to store the lengths of the LCS for all prefixes of the two input sequences.
- **Application:** Let X = (1,2,3,2,4,1,2) and Y = (2,4,3,1,2,1). By applying the dynamic programming algorithm, we can trace back through the computed table to find the subsequence.
  - The subsequence (2,3,2,1) can be found in X as (...,**2**,**3**,**2**,4,**1**,...) and in Y as (**2**,4,**3**,**1**,**2**,**1**).
- **Conclusion:** The length of the LCS is 4. While other common subsequences exist (e.g., 2,4,1,2), the sequence (2,3,2,1) is one of the longest possible, with a length of 4.

Q.54 Which of the followings is NOT a parent selection technique used in genetic algorithm implementations
(1) Radial

(2) Tournament
(3) Boltzmann
(4) Rank

Correct Answer: (1)

Explanation:

- **Concept:** In a Genetic Algorithm (GA), parent selection is the process of choosing individuals from the current population to be parents for the next generation. The goal is to give fitter individuals a higher chance of being selected.
- **Fact (Common Techniques):**
  - **Tournament Selection:** A subset of individuals is chosen randomly, and the fittest one from this group is selected as a parent.
  - **Boltzmann Selection:** Individuals are selected based on a probability distribution derived from their fitness values, using a concept analogous to the Boltzmann distribution in thermodynamics.
  - **Rank Selection:** Individuals are selected based on their rank (from best to worst) rather than their raw fitness score, which helps prevent premature convergence.
- **Conclusion:** Tournament, Boltzmann, and Rank selection are all well-established parent selection methods in GAs. **Radial** is not a standard or recognized parent selection technique in this context.

Q.55 Which of the followings is not a valid property

over two fuzzy relations R and S which must be obeyed for performing α-cut defuzzifications?
(1) $(R\sim \cup S\sim)\lambda = R\sim\lambda \cup S\sim\lambda$
(2) $(R\sim \cap S\sim)\lambda = R\sim\lambda \cap S\sim\lambda$
(3) $(R\sim)\lambda \square = R\sim\lambda$ except when $\lambda=1$
(4) For any $\lambda \leq \beta$, where $0 \leq \beta \leq 1$, it is true that $R\beta \subseteq R\lambda$

Correct Answer: (3)

Explanation:

- **Concept:** An α-cut (or lambda-cut, λ-cut) of a fuzzy set is the crisp set containing all elements whose membership grade in the fuzzy set is greater than or equal to the value α. These properties describe how α-cuts behave under standard fuzzy set operations.
- **Theory:**
  - (1) The α-cut of the union of two fuzzy sets is equal to the union of their individual α-cuts. This is a valid property.
  - (2) The α-cut of the intersection of two fuzzy sets is equal to the intersection of their individual α-cuts. This is also a valid property.
  - (4) This property states that α-cuts are nested. As the threshold (α) increases, the resulting crisp set either shrinks or stays the same. So, the set for a higher threshold (β) is a subset of the set for a lower threshold (λ). This is a fundamental property.
- **Conclusion:** Statement (3) describes the α-cut of the complement of a fuzzy set. The property is generally **not true** that the α-cut of the complement is the complement of the α-cut. The actual

relationship is more complex and depends on whether strong or weak α-cuts are used. Therefore, this is not a valid property that must be obeyed.

Q.56 In a Stop-and-wait system, the bandwidth of the line is 1 Mbps, and 1 bit takes 30 milliseconds to make a round trip. The bandwidth-delay product is
(1) 15,000 bits
(2) 20,000 bits
(3) 30,000 bits
(4) 60,000 bits

Correct Answer: (3)

Explanation:

- **Concept:** The Bandwidth-Delay Product (BDP) is a measure of the number of bits that can be "in transit" on a network link at any given time. It represents the capacity of the link.
- **Formula:** Bandwidth-Delay Product = Bandwidth × Round Trip Time (RTT).
- **Calculation:**
  - Bandwidth = 1 Mbps = 1,000,000 bits per second.
  - The question states that 1 bit takes 30 ms for a round trip. This is a slightly unusual way to phrase it, but it directly gives the RTT. RTT = 30 milliseconds = 0.030 seconds.
  - BDP = 1,000,000 bits/second × 0.030 seconds = 30,000 bits.
- **Conclusion:** The bandwidth-delay product is 30,000 bits. This means that at any point in time, up to

30,000 bits can be on the network link traveling from the sender to the receiver and back.

Q.57 Choose the correct statement for a group G:
(1) If for all x, y ∈ G $(xy)^2=x^2y^2$ then G is Commutative.
(2) If for all x ∈ G, $x^3=1$, then G is Commutative. 1 is the identity element of G.
(3) If for all x ∈ G, $x^5=1$, then G is Commutative. 1 is the identity element of G.
(4) If G is Commutative, the sub-group of G need not be Commutative.

Correct Answer: (1)

Explanation:

- **Concept:** This question tests properties of groups in abstract algebra, specifically the property of commutativity (also known as being an Abelian group). A group is commutative if xy = yx for all elements x, y in the group.
- **Theory (Statement 1):** Let's expand the given condition: $(xy)^2 = x^2y^2$. This means (xy)(xy) = xxyy. Using left cancellation for x and right cancellation for y, we get yx = xy. This is the definition of a commutative group. So, statement (1) is correct.
- **Fact (Statements 2 & 3):** A group where every non-identity element has order 3 (or 5) is not necessarily commutative. There exist non-commutative groups where $x^3=1$ or $x^5=1$ for all x.
- **Fact (Statement 4):** A subgroup of a commutative group is always commutative. If xy = yx for all

elements in the larger group G, it must also be true for all elements in any subset of G, including its subgroups. So, statement (4) is incorrect.

- **Conclusion:** The only statement that is universally true for any group G is the first one.

Q.58 Consider the following interrupt protection levels in Linux, and arrange them in the increasing order of their priorities.
A. User-Mode Programs (Preemptible)
B. Bottom Half Interrupt Handlers
C. Top Half Interrupt Handlers
D. Kernel System Service Routines (Preemptible)
Choose the correct answer from the options given below:
(1) B→A→C→D
(2) C→B→A→D
(3) C→A→B→D
(4) A→D→B→C

Correct Answer: (4)

Explanation:

- **Concept:** In the Linux kernel, different tasks have different levels of urgency or priority. Interrupt handling is the most critical and has the highest priority to ensure the system remains responsive to hardware events.
- **Theory (Hierarchy of Priority):** The priority levels, from lowest to highest, are:

1. **A. User-Mode Programs:** These have the lowest priority and can be preempted by almost any kernel activity.
2. **D. Kernel System Service Routines:** These run in kernel mode on behalf of user programs (e.g., system calls). They have higher priority than user programs but can still be preempted by interrupts.
3. **B. Bottom Halves (Softirqs, Tasklets, etc.):** These are deferrable parts of interrupt handling. They run with interrupts enabled but have higher priority than kernel routines.
4. **C. Top Halves (Hard IRQ Handlers):** These are the immediate, time-critical parts of an interrupt service routine. They run with further interrupts disabled and have the highest priority to service the hardware as quickly as possible.

- **Conclusion:** The correct increasing order of priority is A → D → B → C.

Q.59 Which of the following statement is correct for Pthreads?
(1) It refers to POSIX standard (IEEE 1002.1c)
(2) This standard defines API for thread creation only
(3) This is a specification only for thread behavior and not its implementation
(4) Operating systems like Solaris, Linux & Mac OS X, except Tru64 UNIX, implement Pthreads

Correct Answer: (3)

Explanation:

- **Concept:** Pthreads (POSIX Threads) is a standardized programming interface (API) for creating and managing threads.
- **Fact (Statement 3):** The POSIX standard **specifies the behavior** of the thread library, defining the functions, their parameters, and what they are expected to do. It does **not** dictate how the operating system or library must **implement** this behavior. This allows for different implementations (e.g., user-level or kernel-level threads) on different systems while maintaining source-code portability.
- **Fact (Incorrect Statements):**
  - (1) The correct standard is IEEE 1003.1c (not 1002.1c).
  - (2) The API defines functions for thread creation, synchronization (mutexes, condition variables), termination, and more, not just creation.
  - (4) Tru64 UNIX was one of the systems that did implement the Pthreads standard.
- **Conclusion:** The most accurate statement is that Pthreads is a specification for behavior, leaving the implementation details to the OS vendor.

Q.60 Match List I with List II

List I	**List II**
A. Digital Signature	I. Asymmetric encryption algorithm.
B. Hash Function	II. Confirms authenticity and integrity.
C. AES	III. Produces a fixed-size digest.

D. RSA	IV. Symmetric encryption algorithm.

Choose the correct answer from the options given below:
(1) A-II, B-III, C-I, D-IV
(2) A-II, B-III, C-IV, D-I
(3) A-III, B-II, C-I, D-IV
(4) A-IV, B-II, C-I, D-III

Correct Answer: (2)

Explanation:

- **Concept:** This question tests knowledge of fundamental concepts and algorithms in cryptography.
- **A. Digital Signature:** A digital signature is a cryptographic mechanism used to verify the **authenticity** (proof of origin) and **integrity** (proof that the data has not been altered) of a digital message or document.
- **B. Hash Function:** A cryptographic hash function is an algorithm that takes an arbitrary amount of data as input and **produces a fixed-size string of characters**, which is called a hash value or message digest.
- **C. AES (Advanced Encryption Standard):** AES is a widely used **symmetric encryption algorithm**. This means the same key is used for both encrypting and decrypting the data.

- **D. RSA (Rivest–Shamir–Adleman):** RSA is a widely used **asymmetric encryption algorithm**. This means it uses a pair of keys: a public key for encryption and a private key for decryption.

Q.61 Match List I with List II

List I	**List II**
A. Equivalence Partitioning	I. Measures independent paths in code
B. Boundary Value Analysis	II. Divides input into valid/invalid sets
C. Cyclomatic Complexity	III. Focuses on limits of input ranges
D. Decision Table Testing	IV. In which a number of combinations of actions are tested under varying sets of conditions.

Choose the correct answer from the options given below:
(1) A-II, B-III, C-I, D-IV
(2) A-II, B-III, C-IV, D-I
(3) A-III, B-II, C-I, D-IV
(4) A-IV, B-II, C-I, D-III

Correct Answer: (1)

Explanation:

- **Concept:** This question matches different black-box and white-box software testing techniques with their correct descriptions.

- **A. Equivalence Partitioning:** A black-box testing technique that **divides the input data of a software unit into partitions of equivalent data** from which test cases can be derived. The assumption is that all values in a partition will be processed similarly.
- **B. Boundary Value Analysis:** A black-box testing technique that **focuses on the boundaries or "edges" of input ranges**, as errors often occur at these limits.
- **C. Cyclomatic Complexity:** A white-box testing metric that **measures the number of linearly independent paths** through a program's source code. It indicates the complexity of the code.
- **D. Decision Table Testing:** A black-box testing technique used to test systems with complex business logic. It systematically tests **combinations of input conditions and the resulting actions**.

Q.62 In a relational database, which one of the following is CORRECT:
(1) A relation with only two attributes is always in BCNF.
(2) If all attributes of a relation are prime attributes then the relation is in BCNF.
(3) Every relation has at least one non-prime attribute.
(4) BCNF decomposition preserves functional dependencies.

Correct Answer: (1)

Explanation:

- **Concept:** This question tests the understanding of Boyce-Codd Normal Form (BCNF), a high level of database normalization. A relation is in BCNF if for every non-trivial functional dependency X → Y, X is a superkey.
- **Theory (Statement 1):** Consider a relation R(A, B) with two attributes. The only possible non-trivial functional dependencies are A → B or B → A. In either case, the determinant (A or B) is a candidate key, and thus a superkey. Therefore, any relation with only two attributes is always in BCNF. This statement is correct.
- **Fact (Incorrect Statements):**
  - (2) This is not necessarily true. A relation with all prime attributes can still have a dependency where the determinant is not a superkey (e.g., if the key is {A, B} and a dependency C → D exists, where C is a prime attribute but not a superkey).
  - (3) A relation can have all its attributes as part of the candidate key (all prime attributes).
  - (4) BCNF decomposition is not always dependency-preserving. This is a known limitation compared to 3NF.
- **Conclusion:** The only statement that is always true is the first one.

Q.63 Match List I with List II

List I	**List II**
A. Overloading	I. Since function call is resolved during run time, the execution is Slow.

B. Early binding	II. Since function call is resolved during compilation time, the execution is much faster.
C. Overriding	III. Supports compile-time polymorphism.
D. Late Binding	IV. Supports run-time polymorphism.

Choose the correct answer from the options given below:
(1) A-III, B-II, C-I, D-IV
(2) A-IV, B-II, C-III, D-I
(3) A-III, B-I, C-IV, D-II
(4) A-IV, B-I, C-III, D-II

Correct Answer: (1) (Note: The provided answer key 'D' is invalid. The correct match is (1)).

Explanation:

- **Concept:** This question matches key concepts of Object-Oriented Programming (OOP), specifically related to polymorphism and binding.
- **A. Overloading (Function Overloading):** This allows multiple functions with the same name but different parameters. The correct function to call is determined at compile-time based on the arguments. It is a form of **compile-time polymorphism**.
- **B. Early Binding (Static Binding):** The process of linking a function call to the function definition

during **compilation time**. This is efficient and results in **faster execution**.

- **C. Overriding (Function Overriding):** This allows a subclass to provide a specific implementation of a method that is already provided by its superclass. It is a key feature of **run-time polymorphism**. Because the decision is made at runtime (via virtual functions), it can lead to **slower execution** compared to early binding.
- **D. Late Binding (Dynamic Binding):** The process of linking a function call to the function definition during **run time**. This is what enables **run-time polymorphism**.

Q.64 Arrange the following in the increasing order of coupling from lowest coupling to highest coupling.
A. Common Coupling
B. Stamp Coupling
C. Control Coupling
D. External Coupling
E. Content Coupling
Choose the correct answer from the options given below:
(1) E, A, C, B, D
(2) D, B, A, E, C
(3) B, C, D, A, E
(4) C, A, B, D, E

Correct Answer: (3)

Explanation:

- **Concept:** In software engineering, coupling is the degree of interdependence between software modules. Low coupling is desirable as it means a change in one module will have less impact on others. The order from lowest (best) to highest (worst) is a standard concept.
- **Theory (The Hierarchy):** The standard hierarchy of coupling from lowest to highest is:
  1. Data Coupling (not listed)
  2. **Stamp Coupling (B):** Modules share a composite data structure but each use different parts of it.
  3. **Control Coupling (C):** One module passes a flag or command to control the logic of another.
  4. **External Coupling (D):** Modules share a dependency on an external format, protocol, or device.
  5. **Common Coupling (A):** Modules share global data.
  6. **Content Coupling (E):** One module directly modifies or relies on the internal workings of another module.
- **Conclusion:** Based on the options provided, the correct increasing order of coupling (from best to worst) is B → C → D → A → E.

Q.65 The write operation in I/O operation does the following-
(1) Transfer data from I/O device to memory
(2) Transfer data from memory to I/O device
(3) Transfer data from CPU register to memory
(4) Transfer data from CPU register to I/O device

Correct Answer: (2)

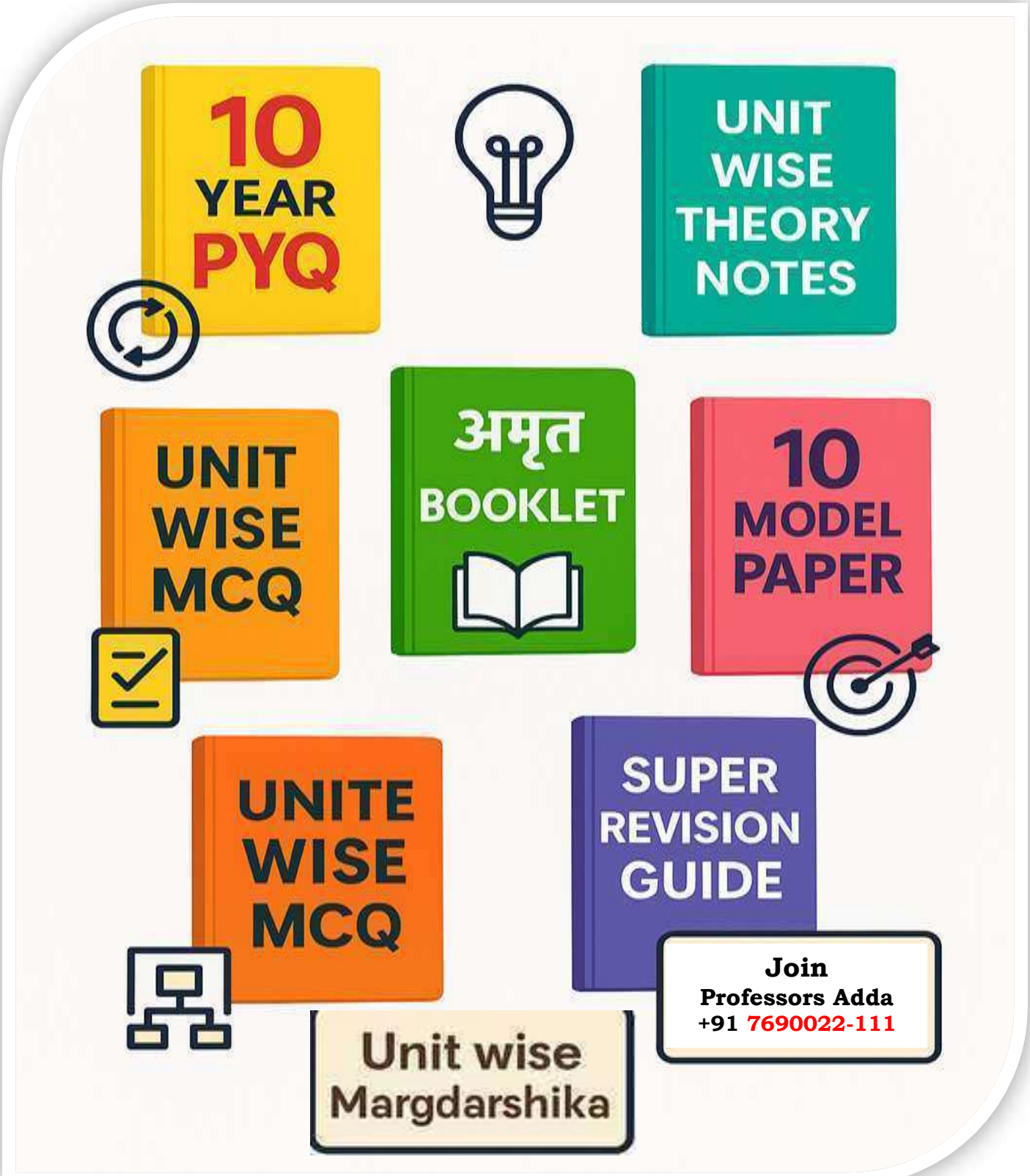
Explanation:

- **Concept:** This question asks for the definition of a "write" operation in the context of Input/Output (I/O) from the perspective of the computer's central processing unit (CPU) and main memory.
- **Theory:**
  - A **read** operation involves bringing data *into* the main system from an external device (e.g., reading a file from a hard disk into memory).
  - A **write** operation involves sending data *out* from the main system to an external device (e.g., writing data from memory to a file on a hard disk, or sending data to a printer).
- **Fact:** The flow of data in a write operation is from the computer's internal storage (main memory) to the peripheral I/O device.
- **Conclusion:** Therefore, a write operation transfers data from memory to an I/O device.

UGC NET / JRF / A. Professor / CUET

हिंदी English माध्यम उपलब्ध

Paper 1 & All Subject Available



CS OPERATING SYSTEM
MULTIPLE CHOICE QUESTIONS

1. What is cache-coherency?

(a) Ensuring data in main memory matches data on the disk.
(b) Maintaining consistency of data stored in multiple local caches within a multiprocessor system.
(c) Verifying the integrity of data transmitted over a network.
(d) Optimizing cache usage to minimize miss rates.

Correct Answer: (b)

Explanation:

- In multiprocessor systems, each processor often has its own local cache memory for faster data access.
- If multiple processors cache the same block of main memory, inconsistencies can arise if one processor modifies its cached copy.
- Cache coherence protocols ensure that any change to a shared data item in one cache is propagated to other caches holding that item.
- This guarantees that all processors see a consistent view of memory.
- Common protocols include snooping (caches monitor bus traffic) and directory-based (a central directory tracks shared data).
- Maintaining coherence is crucial for the correctness of parallel programs.

2. What is deadlock prevention techniques?
(a) Using faster hardware, Increasing memory, Optimizing code.
(b) Regularly restarting the system, Killing blocked processes manually.
(c) Implementing algorithms to detect and recover from deadlocks after they occur.
(d) Ensuring that at least one of the four necessary conditions for deadlock cannot hold.

Correct Answer: (d)
Explanation:
- Deadlock prevention aims to design the system in such a way that deadlocks are structurally impossible.
- This is achieved by constraining how resource requests can be made, ensuring that one of the four necessary conditions is never met:
  - Negating Mutual Exclusion: Make resources sharable (not always possible).
  - Negating Hold and Wait: Require processes to request all resources at once, or release all held resources before requesting new ones.
  - Negating No Preemption: Allow resources to be preempted (forcibly taken away) from waiting processes.
  - Negating Circular Wait: Impose a total ordering on resource types and require processes to request resources in increasing order.
- Prevention methods can lead to low resource utilization or potential starvation.

3. What is a P-thread?
(a) A type of kernel thread used in Windows.
(b) A proprietary threading library from Sun Microsystems.
(c) A POSIX standard (IEEE 1003.1c) API for thread creation and synchronization.
(d) A physical thread of execution on a multi-core processor.

Correct Answer: (c)
Explanation:
- Pthreads stands for POSIX Threads.
- It is not an implementation itself, but rather a specification (an API standard) defined by IEEE POSIX 1003.1c.
- It defines a set of C language programming types and procedure calls for creating, managing, and synchronizing threads.
- Implementations of the Pthreads API are widely available on UNIX-like operating systems (Linux, macOS, Solaris, etc.).
- It provides a portable way to write multithreaded applications across different platforms that support the standard.

4. What is a job queue?
(a) A queue of processes currently running on the CPU.
(b) A queue of processes waiting for a specific I/O device.
(c) The set of all processes in the system, including those waiting to be admitted to memory.
(d) A queue of processes that have terminated but are waiting to be cleaned up.

Correct Answer: (c)

Explanation:

- The job queue represents the collection of all processes that have entered the system.
- It includes processes that might still reside on secondary storage (disk).
- The long-term scheduler (or job scheduler) selects processes from this queue to be loaded into main memory for execution.
- It encompasses processes in various states, forming the pool from which the OS selects candidates for execution.
- In simple systems, it might just be the set of processes submitted for batch processing.

5. What is a ready queue?

(a) The set of all processes currently loaded in main memory.
(b) The list of processes residing in main memory, ready and waiting to execute.
(c) The queue of processes waiting for I/O operations to complete.
(d) The queue of processes selected by the long-term scheduler.

Correct Answer: (b)

Explanation:

- The ready queue holds processes that are fully prepared to run on the CPU.
- These processes are already loaded into main memory.
- They possess all necessary resources except the CPU itself.
- The short-term scheduler (or CPU scheduler) selects the next process to run from the ready queue.
- Processes enter the ready queue when they are first admitted, when they finish waiting for an event (like I/O), or when preempted from the CPU.

6. What are the benefits of multithreaded programming?

(a) Increased security, simplified code, reduced memory usage.
(b) Guaranteed faster execution, automatic error correction.
(c) Responsiveness, Resource sharing, Economy, Utilization of multiprocessor architectures.
(d) Lower hardware requirements, easier debugging.

Correct Answer: (c)

Explanation:

- Responsiveness: Allows an application to remain responsive to user input even if part of it is blocked or performing a long operation (e.g., UI thread remains active).
- Resource Sharing: Threads within a process share memory and resources by default, simplifying data sharing compared to IPC between processes.
- Economy: Creating and context-switching threads is generally less resource-intensive than for processes.

- Utilization of Multiprocessor Architectures: Multiple threads can run in parallel on different CPU cores, significantly increasing performance for CPU-bound tasks.

7. What is a layered approach and what is its advantage?

(a) A hardware design with multiple circuit layers; advantage is smaller size.
(b) Structuring the OS into levels, each built on the one below it; advantage is modularity and easier debugging.
(c) A file system with multiple layers of directories; advantage is organization.
(d) A networking model with different protocol layers; advantage is standardized communication.

Correct Answer: (b)

Explanation:

- The layered approach organizes the OS into a hierarchy of layers (levels).
- The lowest layer (Layer 0) is the hardware; the highest (Layer N) is the user interface.
- Each layer implements a set of functions (operations) that can only use functions provided by lower-level layers.
- This structure promotes modularity, as layers hide implementation details from higher layers.
- Debugging and verification are simplified because errors can be localized to specific layers.
- A potential disadvantage is reduced performance due to the overhead of passing requests through multiple layers.

8. What is a Real-Time System?

(a) A system that provides real-time stock market data.
(b) A system where processing speed is the only priority.
(c) An operating system and associated hardware/software where processing must respond to events within specific time constraints.
(d) A system designed for realistic simulations and graphics.

Correct Answer: (c)

Explanation:

- Real-time systems are defined by their timing requirements, not just speed.
- The correctness of the system depends on both the logical result and the time at which it's produced.
- They often interact with the physical world through sensors and actuators.
- Used in applications like industrial control, robotics, medical imaging, flight control systems.
- Require predictable performance and timely responses to external events.
- Utilize Real-Time Operating Systems (RTOS) designed to meet these constraints.

9. What are types of threads?
(a) Parent threads and Child threads.
(b) Foreground threads and Background threads.
(c) User threads and Kernel threads.
(d) Synchronous threads and Asynchronous threads.

Correct Answer: (c)
Explanation:
- User Threads: Managed entirely in user space by a thread library without kernel support. Creation and switching are fast. However, if one user thread blocks on a system call, the entire process might block, even if other threads are ready. Cannot take advantage of multiple processors.
- Kernel Threads: Supported and managed directly by the operating system kernel. Creation and switching are slower than user threads. If one kernel thread blocks, the kernel can schedule another thread (from the same or different process). Can run in parallel on multiprocessors.
- Hybrid approaches (like many-to-many models) also exist.

10. Explain the concept of the multi-programmed operating systems?
(a) Multiple physical processors execute different programs simultaneously.
(b) A single processor executes only one program from start to finish before starting the next.
(c) Multiple programs reside in memory concurrently, and the CPU switches between them when one waits for I/O.
(d) Each user gets a dedicated time slice to run their program exclusively.

Correct Answer: (c)
Explanation:
- Multiprogramming aims to keep the CPU busy by overlapping CPU execution with I/O operations.
- Several jobs are loaded into main memory (job pool/ready queue).
- The OS selects one job to run (job scheduling/CPU scheduling).
- If the running job needs to perform I/O (a slow operation), the OS doesn't wait; it switches the CPU to another ready job.
- When the I/O for the first job completes, it returns to the ready state, waiting for the CPU again.
- This concurrent (not simultaneous on a single CPU) execution increases overall system throughput.

11. What is a deadlock?
(a) A situation where a process crashes and cannot be restarted.
(b) A security vulnerability allowing unauthorized access.
(c) A state where two or more processes are blocked indefinitely, each waiting for a resource held by another process in the set.
(d) A condition where the CPU is overloaded with too many processes.

Correct Answer: (c)

Explanation:

- Deadlock is a specific type of blocking scenario in concurrent systems.
- It occurs when a set of processes are all waiting for an event (typically resource acquisition) that can only be caused by another process in the same set.
- Since all processes in the set are waiting, none of them can trigger the event needed to unblock others.
- This results in a permanent standstill for the involved processes.
- Deadlocks arise due to competition for shared resources under specific conditions.

12. What is dining philosophers' problem?

(a) A resource allocation problem in distributed systems.
(b) A classic synchronization problem illustrating deadlock and starvation using philosophers, chopsticks, and rice.
(c) A scheduling problem for tasks with dependencies.
(d) A game theory problem involving resource competition.

Correct Answer: (b)

Explanation:

- This problem, posed by Dijkstra, models processes competing for limited resources.
- Five philosophers sit around a table, alternating between thinking and eating. Five chopsticks are placed between them.
- To eat, a philosopher needs two chopsticks (the ones to their left and right). They can only pick up one chopstick at a time.
- The problem is to design a protocol (synchronization mechanism) that allows philosophers to eat without deadlock (e.g., all picking up their left chopstick simultaneously and waiting forever for the right) and without starvation (one philosopher never getting to eat).
- It's used to test and compare different synchronization schemes.

13. What is the purpose of different operating systems?

(a) All operating systems primarily focus on optimizing hardware utilization.
(b) Workstations prioritize hardware optimization, while mainframes focus on usability.
(c) Mainframes optimize hardware use; Workstations focus on individual usability; Handhelds prioritize ease of use and low power.
(d) PCs are designed only for games, while handhelds are for business applications.

Correct Answer: (c)

Explanation:

- Different computing environments have different needs, leading to specialized OS designs.
- Mainframe OS aim to maximize the utilization of expensive hardware resources across many tasks.
- Workstation/PC OS focus on providing a responsive and feature-rich environment for individual users and applications (like games, business software).
- Handheld (mobile) OS prioritize a simple user interface, power efficiency for battery life, and connectivity.
- The purpose dictates the design trade-offs, such as performance vs. power consumption or complexity vs. ease of use.
- Understanding these purposes helps explain the features and limitations of various OS types.

14. What is a semaphore?

(a) A hardware signal used for inter-process communication.
(b) A variable used to count the number of active processes.
(c) A synchronization tool; an integer variable accessed only via atomic wait (P) and signal (V) operations.
(d) A type of memory barrier ensuring instruction ordering.

Correct Answer: (c)

Explanation:

- Semaphores are a classic synchronization primitive invented by Edsger Dijkstra.
- A semaphore S is an integer variable.
- It can only be accessed through two standard, atomic operations:
  - wait(S) (or P(S)): Decrements the semaphore value. If the value becomes negative, the process blocks (waits). wait(S) { while (S <= 0) ; S--; } (Conceptual definition; actual implementation avoids busy waiting).
  - signal(S) (or V(S)): Increments the semaphore value. If there are processes waiting on S, one is unblocked. signal(S) { S++; }
- Counting semaphores can range over an unrestricted domain. Binary semaphores (mutexes) can only be 0 or 1 and are used for mutual exclusion.

15. What is the difference between Hard and Soft real-time systems?

(a) Hard systems use hardware timers; Soft systems use software timers.
(b) Hard systems have fixed deadlines that must be met; Soft systems prioritize tasks but tolerate occasional deadline misses.
(c) Hard systems run on dedicated hardware; Soft systems run on general-purpose computers.
(d) Hard systems are complex; Soft systems are simple.

Correct Answer: (b)

Explanation:

- Hard Real-Time System: Missing a deadline constitutes a total system failure. Guarantees are required. Examples: Flight control systems, anti-lock brakes. Requires careful design and analysis to ensure all deadlines are met under all conditions.
- Soft Real-Time System: Missing a deadline degrades performance but doesn't cause system failure. Critical tasks are prioritized over non-critical ones. Examples: Multimedia streaming (slight delays are acceptable), online transaction systems. Allows more flexibility in design.

16. What are the deadlock avoidance algorithms?

(a) Algorithms that prevent deadlocks by breaking one of the four necessary conditions.
(b) Algorithms that detect deadlocks after they occur and then recover.
(c) Algorithms that dynamically check the resource allocation state to ensure the system never enters an unsafe state (e.g., Banker's algorithm).
(d) Algorithms for scheduling processes to avoid resource conflicts.

Correct Answer: (c)

Explanation:

- Deadlock avoidance requires the OS to have advance information about the maximum resource needs of each process.
- When a process requests a resource, the avoidance algorithm checks if granting the request would leave the system in a safe state.
- If granting the request leads to a safe state, the resource is allocated.
- If granting the request would lead to an unsafe state (potential for deadlock), the process must wait, even if the resource is currently available.
- The Banker's algorithm is a well-known deadlock avoidance algorithm for systems with multiple instances of resource types. The Resource Allocation Graph algorithm can be used for single-instance resource types.

17. What are the different operating systems?

(a) Windows, macOS, Linux, Android, iOS
(b) Batched, Multi-programmed, Timesharing, Distributed, Real-time
(c) Single-user, Multi-user, Networked, Embedded, Cloud
(d) Kernel-based, Microkernel-based, Monolithic, Layered, Virtualized

Correct Answer: (b)

Explanation:

- This question refers to major classifications based on how the OS manages jobs and interacts with users/hardware.
- Batched systems processed jobs sequentially in groups without user interaction.
- Multi-programmed systems kept multiple jobs in memory, switching between them to maximize CPU use.
- Timesharing extended multiprogramming to allow interactive user sessions by rapidly switching between users.

- Distributed systems manage resources across multiple, independent computers connected via a network.
- Real-time systems are designed to process data and events within strict time constraints, crucial for control systems.

18. What is a sector?

(a) A section of main memory allocated to a process.
(b) The smallest addressable unit of data storage on a magnetic disk or optical disc.
(c) A component of the CPU responsible for arithmetic operations.
(d) A type of network packet.

Correct Answer: (b)

Explanation:

- Disk drives organize data physically into tracks (concentric circles) and sectors (segments of a track).
- A sector is the minimum amount of data that can be read from or written to the disk surface in a single operation.
- Historically, sector sizes were often 512 bytes, but modern drives often use larger sectors (e.g., 4096 bytes or 4K).
- The operating system's file system maps logical blocks (its unit of allocation) to these physical sectors.
- Addressing data involves specifying the cylinder (track position), head (surface), and sector number.

19. What is Throughput, Turnaround time, waiting time and Response time?

(a) Metrics related to network performance.
(b) Measures of disk I/O speed and efficiency.
(c) Criteria used to evaluate CPU scheduling algorithm performance.
(d) Parameters describing memory access speeds.

Correct Answer: (c)

Explanation:

- These are common metrics for analyzing scheduling algorithms:
  - Throughput: Number of processes completed per unit of time. (Maximize)
  - Turnaround Time: Total time taken for a particular process to execute, from submission to completion (waiting time + execution time + I/O time). (Minimize)
  - Waiting Time: Total time a process spends waiting in the ready queue. (Minimize)
  - Response Time: Time from when a request was submitted until the first response is produced (not necessarily full output). Crucial for interactive systems. (Minimize)
- Different algorithms optimize for different criteria.

Computer Networks

This section provides information on various calculations and protocols related to computer networks, including:

- **Delay Calculations:** Formulas for calculating Propagation Delay and Transmission Delay.
- **Channel Utilization:** Formulas for channel utilization for various protocols (e.g., Ethernet, Token Ring, Stop-and-Wait, Sliding Window).
- **Signal and Noise Calculations:** Calculation of Signal to Noise Ratio and Signal Attenuation.
- **Data Rate and Channel Capacity Calculations:** Formulas based on Nyquist Theorem and Shannon's theorem.
- **MAC Sublayer:** Static and Dynamic Channel Allocation.
- **Multiple access protocol:** Explanation of Pure ALOHA and Slotted ALOHA protocols and throughput calculations.
- **ROUTING ALGORITHMS:** Various routing algorithms (Static and Adaptive).
- **Traffic Shaping:** Leaky Bucket and Token Bucket algorithms.
- **Datagram format:** Various fields of an IP datagram and their lengths.
- **IP class addressing:** Different IP classes (A, B, C, D, E) and their ranges.
- **Transport Layer:** TCP RTT calculation and comparison with UDP.
- **CLIENT SERVER MODEL:** Socket programming steps for client and server sides.
- **Sliding Window Protocols:** Sender and Receiver window sizes for Go-Back-N and Selective Repeat protocols.
- **Devices and their OSI Layer:** Different network devices and the OSI layer they operate on.

COMPUTER NETWORKS

Delay Calculations:

- Propagation delay = $\frac{Distance\ Traveled\ by\ frame\ or\ packet}{Propagaton\ speed}$
- Transmission delay = $\frac{No.of bits\ to\ transfer}{Transmission\ speed\ or\ bandwidth(in\ bps)}$

Channel Utilization:

1. For *Ethernet*, channel utilization, U = $\frac{1}{1+5a}$, where a = $\frac{propagation\ delay}{transmission\ delay}$
   1.1. *for 1- persistent CSMA/CD* LAN 10-1000Mbps speed, simplified channel efficiency is,
   $[Channel\ efficiency] = \frac{T_d}{T_d+\frac{2\tau}{A}}$, where A =$kp(1-p)^{k-1}$, k stations each with probability p to transmit in a contention slot. T_d is the time to transmit a frame. τ is the worst case one way propagation delay.
   1.2. *Efficiency of 802.3 Ethernet at 10Mbps with 512 bit connection*:
   $(Channel\ Efficiency) = \frac{1}{1+\frac{2BLe}{cF}}$, where, B = network bandwidth. L = Cable Length. C = speed of signal propagation. E = optimal number if contention slots per frame. F = frame size
2. For *Token ring* (**release after transmission or early token release**),
   2.1. Channel utilization, U =$\frac{1}{1+\frac{a}{N}}$, where N is the number of stations in the ring.
3. For *Token ring* (**release after reception or delayed token release**),
   3.1. Channel utilization, U =$\frac{1}{1+a}$, where N is the number of stations in the ring.
4. For *unrestricted simplex protocol*: If a frame has d data bits and h overhead bits and channel bandwidth = b bits/sec then,
   4.1. Maximum channel utilization = $\frac{Data\ Size}{Frame\ Size}$ = $\frac{d}{d+h}$
   4.2. Maximum data throughput = $\frac{Data\ Size}{Frame\ Size} \times Bandwidth$ = $\frac{d}{d+h} \times b$
5. For *stop-and-wait*,
   5.1. Channel utilization, U =$\frac{1-p}{1+2a}$, where a = $\frac{propagation\ delay}{transmission\ delay}$ and p is the probability that a frame is in error.
   5.2. Also Maximum channel utilization = $\frac{Time\ to\ transmit\ a\ frame}{Round\ trip\ time\ (R)} \times \frac{d}{d+h}$ = $\frac{d}{b \times R}$

5.3. Maximum data throughput = $\frac{d}{b \times R} \times b = \frac{d}{R}$

6. For Simplex positive acknowledgement with retransmission (PAR) protocol: -
   6.1. Maximum channel utilization and throughput is similar to stop-and-wait protocol when the effect of errors is ignored.
7. For *Sliding Window Protocols* with window size of **w**,
   7.1. Go-Back-N,
      7.1.1. Channel utilization, u

$$= \begin{cases} \frac{1-p}{1+2ap}, & if\ window\ fills\ the\ pipe\ i.e., w \geq 2a+1 \\ \frac{w(1-p)}{(1+2a)(1-p+wp)}, & if\ window\ does\ not\ fills\ the\ pipe\ i.e., w < 2a+1 \end{cases}$$

   7.2. Selective reject,

7.2.1. Channel utilization ,u $= \begin{cases} (1-p), & if\ window\ fills\ the\ pipe\ i.e., w \geq 2a+1 \\ \frac{w(1-p)}{(1+2a)}, & if\ window\ does\ not\ fills\ the\ pipe\ i.e., w < 2a+1 \end{cases}$

   7.3. Condition for maximum utilization or throughput is:

$$[Time\ to\ transmit\ \boldsymbol{w}\ frames] \geq [Round\ Trip\ Time]$$

Throughput Calculations:

Throughput = Channel Utilization × Channel Bandwidth

Signal and Noise Calculations:

1. *Signal to Noise Ratio* (in decibels, **dB**) = $10\log_{10}\frac{S}{N}$,
   a. where S= Signal strength and N = noise strength.
2. *Signal Attenuation* (in decibels, **dB**) = $10\log_{10}\frac{Transmitted\ Power}{Received\ Power}$,

Data Rate and Channel Capacity Calculations:

1. *Nyquist Theorem*: Maximum data rate = $2H\log_2 V$ bits/sec, where H is bandwidth in hertz (Hz) and V is number of levels.
2. *Shannon's theorem*: Channel capacity = $H\log_2\left(1+\frac{S}{N}\right)$ bits/sec, where H is bandwidth in hertz (Hz). (Note: here $\frac{S}{N}$ is **not** in decibels).

Baud rate: A baud is the number of changes per second in the signal.

- For Manchester encoding, **baud rate = 2 × bit-rate**

MAC Sub layer:

Static channel allocation in LANs and MANs.

If C = channel capacity in bps

λ = arrival rate of frames (frames/sec)

$\frac{1}{\mu}$ = no. of bits per frame, then

Mean time to delay, T = $\frac{1}{\mu C - \lambda}$,

Now, if the channel is divided into N sub channels each with capacity $\frac{C}{N}$ and arrival rate or input rate on each of the N channels is $\frac{\lambda}{N}$ then,

T'(fdm) = $\frac{1}{\mu\frac{C}{N} - \frac{\lambda}{N}} = \frac{N}{\mu C - \lambda}$

Dynamic Channel Allocation:

$\begin{bmatrix} Probability\ of\ a\ frame \\ being\ generated\ in\ a\ period\ of\ length\ \Delta t \end{bmatrix} = \lambda \Delta t$, where λ is the arrival rate of frames.

Multiple access protocol:

Pure ALLOHA protocol

- Infinite senders assumed.
- Poisson distribution with mean rate **S** frames per frame time
- Combined frame rate with retransmission **G** frames per frame time.
- '**t**' is the time required to transmit a frame.
- In multiple access protocol, a frame is successful if no other frames are transmitted in the vulnerable period.
- Probability of **k** frames being generated during a frame transmission time:
  $P_k = \frac{G^k e^{-G}}{k!}$
- Hence, probability of zero frames in 2 frame periods is, $P_0 = e^{-2G}$
- Therefore, for pure ALLOHA,
  - Mean rate $S = GP_0 = Ge^{-2G}$ which becomes maximum at G = ½,
  Max(S) = $\frac{1}{2e}$ = 0.184 = 18.4% throughput.
- Vulnerability period in pure ALLOHA: For successful frame transmission, no other frame should be on the channel for vulnerability period equal to twice the time to transmit one frame. That is,
  $\begin{pmatrix} Vulnerability\ period \\ in\ case\ of\ PURE\ ALLOHA \end{pmatrix} = 2t$, where t is the time to transmit one frame.

Slotted ALLOHA protocol

- Time is divided into discrete frame slots.
- A station is required to wait for the beginning of the next slot to transmit a frame.
- Vulnerability period is halved as opposed to pure ALLOHA protocol. That is,
  $\begin{pmatrix} Vulnerability\ period \\ in\ case\ of\ SLOTTED\ ALLOHA \end{pmatrix} = t$, where t is the time to transmit one frame.
- Probability of **k** frames being generated during a frame transmission time:
  $P_k = e^{-G}(1 - e^{-G})^{k-1}$

PROFESSORS ADDA

Trusted By Toppers

UGC-NET CSIR PGT SET CUET JRF ASST. PROF

STUDY KIT

COD

SPECIAL PRICE

REGULAR PRICE

~~₹ 5999~~

LIMITED TIME PRICE

₹ ***

click here

+91 7690022111 +91 9216228788

TESTIMONIALS

Nikita Sharma
UGC NET (PAPER 1)
Delhi

"The premium course by Professors Adda gave me everything in one place – structured notes, MCQ banks, PYQs, and trend analysis. The way it was aligned with the syllabus helped me stay organized and confident."

Ravindra Yadav
UGC NET (Commerce)
Jaipur

"Joining the premium group was the best decision I made. The daily quiz challenges, mentor guidance, and focused discussions kept me disciplined and exam-ready."

Priya Mehta
UGC NET (Education)
Banglore

"Professors Adda's study course is like a personal roadmap to success. The live sessions and targeted revision plans were crucial in helping me clear my exam on the first attempt."

Swati Verma
UGC NET (English Literature)
Kolkata

"What makes the Professors Adda premium course unique is the combination of high-quality content and a dedicated support group. It kept me motivated and accountable throughout."

Aman Joshi
UGC NET (Sociology)
Prajagraj

"The premium group gave me access to serious aspirants and mentors who guided me every step of the way. The peer learning, doubt sessions, and motivation from the group were unmatched."

Riya Sharma
UGC NET (Psychology)
Hyderabad

"What really kept me going was the constant encouragement from Professors Adda's mentors. Their support helped me stay motivated even when I felt overwhelmed by the syllabus."

Anjali Singh
UGC NET (Political Science)
Indore

"Professors Adda taught me that smart preparation is as important as hard work. Their strategic study plans and motivational talks made all the difference in my success."

Aditya Verma
UGC NET (History)
Guwahati

"The institute not only provides excellent study resources but also builds your confidence. The motivational sessions helped me overcome exam anxiety and keep a positive mindset."

*IMAGES ARE IMAGINARY

+91 7690022111+91 9216228788

PROFESSORS ADDA

Trusted By Toppers

GET BEST SELLER HARD COPY NOTES

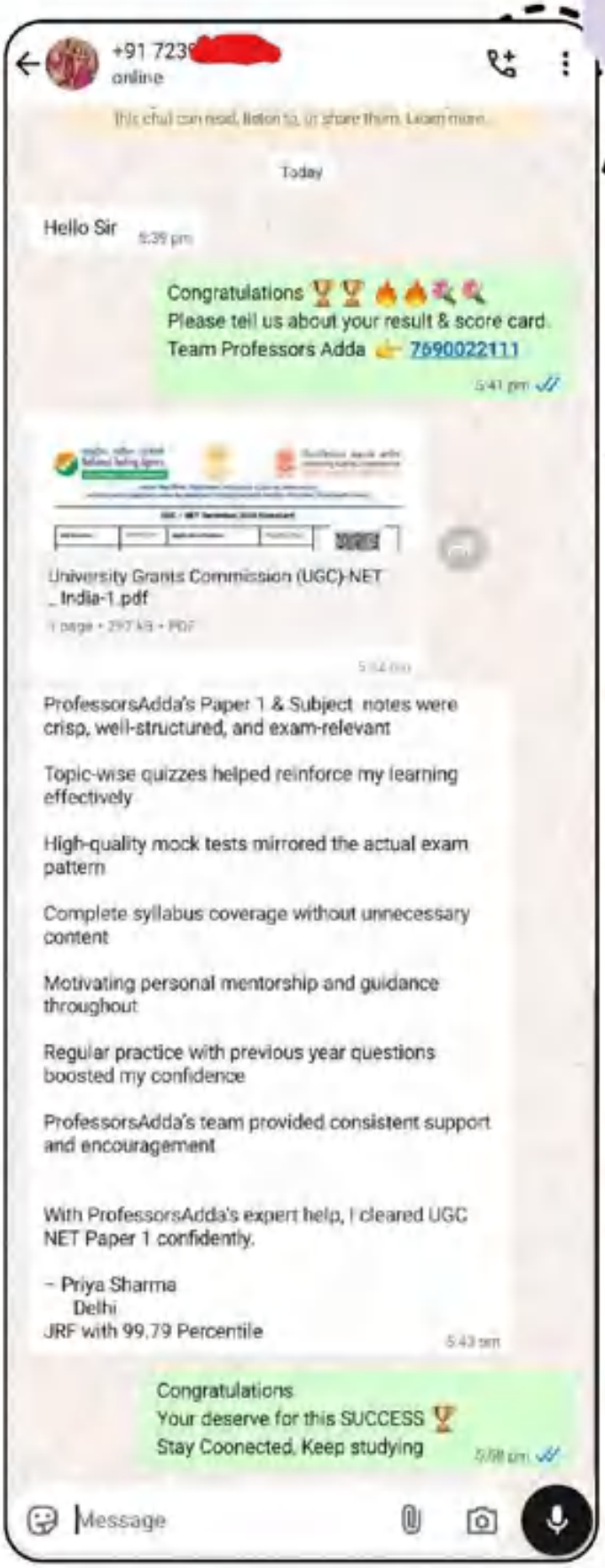
CLICK HERE TO GET

+91 7690022111+91 9216228788

PROFESSORS ADDA

Trusted By Toppers

Our Toppers



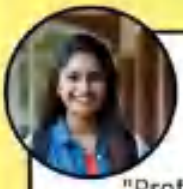
TESTIMONIALS

Nikita Sharma
UGC NET (PAPER 1)
Delhi

"The premium course by Professors Adda gave me everything in one place – structured notes, MCQ banks, PYQs, and trend analysis. The way it was aligned with the syllabus helped me stay organized and confident."

Ravindra Yadav
UGC NET (PAPER 1)
Jaipur

"Joining the premium group was the best decision I made. The daily quiz challenges, mentor guidance, and focused discussions kept me disciplined and exam-ready."



Priya Mehta
UGC NET (PAPER 1)
Banglore

"Professors Adda's study course is like a personal roadmap to success. The live sessions and targeted revision plans were crucial in helping me clear my exam on the first attempt."

Swati Verma
UGC NET (PAPER 1)
Kolkata

"What makes the Professors Adda premium course unique is the combination of high-quality content and a dedicated support group. It kept me motivated and accountable throughout."

Aman Joshi
UGC NET (PAPER 1)
Prajagraj

"The premium group gave me access to serious aspirants and mentors who guided me every step of the way. The peer learning, doubt sessions, and motivation from the group were unmatched."

Riya Sharma
UGC NET (PAPER 1)
Hyderabad

"What really kept me going was the constant encouragement from Professors Adda's mentors. Their support helped me stay motivated even when I felt overwhelmed by the syllabus."

Anjali Singh
UGC NET (PAPER 1)
Indore

"Professors Adda taught me that smart preparation is as important as hard work. Their strategic study plans and motivational talks made all the difference in my success."

Aditya Verma
UGC NET (PAPER 1)
Guwahati

"The institute not only provides excellent study resources but also builds your confidence. The motivational sessions helped me overcome exam anxiety and keep a positive mindset."

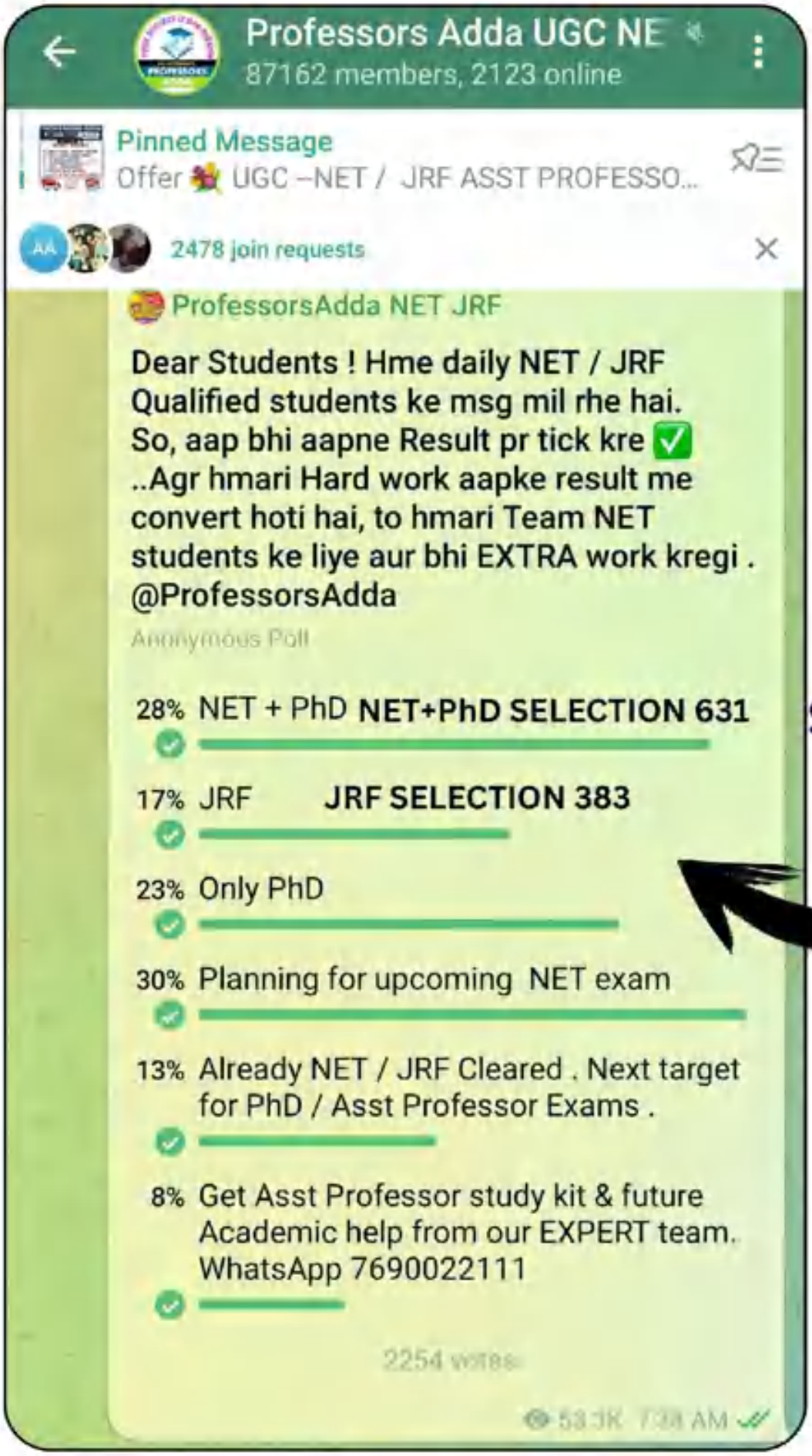
*IMAGES ARE IMAGINARY

+91 7690022111+91 9216228788

PROFESSORS ADDA

Trusted By Toppers

Our Toppers



OUR UGC NET SELECTION RESULTS

PROFESSORS ADDA

Trusted By Toppers

Exclusive English GROUP

INDIA'S NO 1 UGC NET GROUP

CLICK HERE TO JOIN

+91 7690022111 +91 9216228788

PROFESSORS ADDA

Trusted By Toppers

OUR UGC NET SELECTION RESULTS

MANY MORE SELECTION

+91 7690022111+91 9216228788

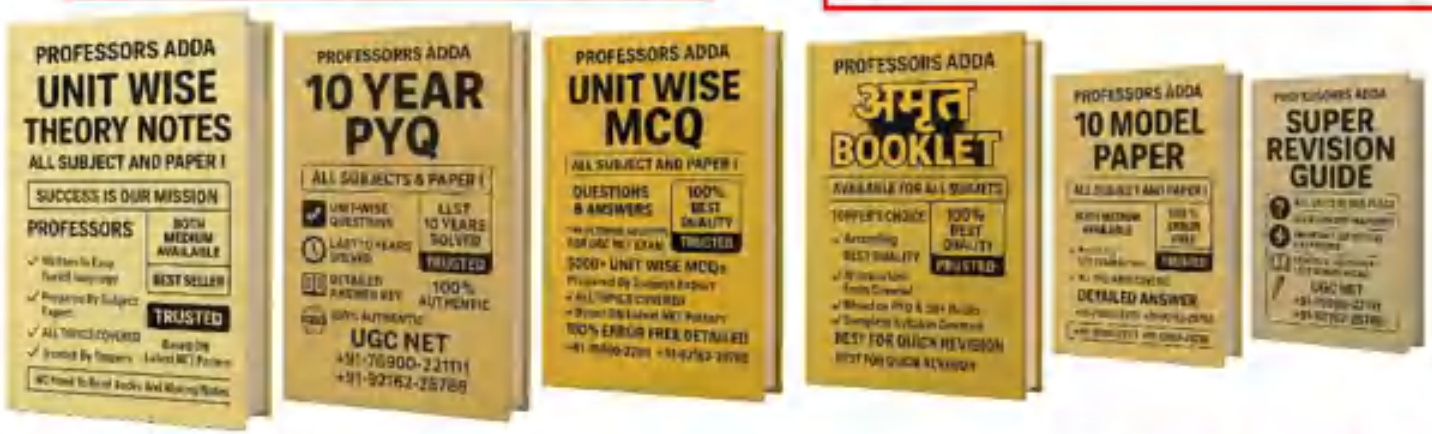
PROFESSORS ADDA

Trusted By Toppers

BOOK YOUR HARD COPY
COMPLETE STUDY PACKAGE

Hurry! Limited copies remaining—get yours before they're gone.

- 10 Unit Theory Notes
- Unit Wise MCQ Bank
- Latest 10 YEAR PYQ
- Model Papers
- One Liner Quick Revision Notes
- Premium Group Membership



FREE sample Notes/
Expert Guidance /Courier Facility Available

Download PROFESSORS ADDA APP

91-76900-22111

PROFESSORS ADDA

Trusted By Toppers

BOOK YOUR HARD COPY
COMPLETE STUDY PACKAGE

Hurry! Limited copies remaining—get yours before they're gone.

NEW PRODUCT

- 10 Unit Theory Notes
- Unit Wise MCQ Bank
- Latest PYQ
- Model Papers
- One Liner Quick Revision Notes
- Premium Group Membership

FREE sample Notes/
Expert Guidance /Courier Facility Available

Download PROFESSORS ADDA APP

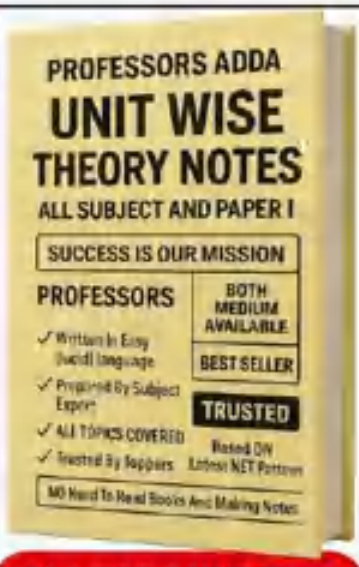
91-76900-22111

OUR ALL PRODUCTS

NEW PRODUCT

CLICK HERE

NEW PRODUCT



CLICK HERE

NEW PRODUCT

CLICK HERE

NEW PRODUCT

CLICK HERE

NEW PRODUCT

CLICK HERE

NEW PRODUCT

CLICK HERE

+91 7690022111+91 9216228788