UGC NET Computer Science 2nd January 2026 Memory Based Paper Shift - 1

Q.1 Which of the following is a tautology in propositional logic?

A. $(p \wedge \neg p)$
B. $(p \vee \neg p)$
C. $(p \rightarrow q) \wedge (\neg q \rightarrow \neg p)$
D. $(\neg p \vee p) \wedge (q \rightarrow q)$

Answer: B

Sol: A **tautology** is a logical statement that is **always true**, regardless of the truth values of its components. Let's evaluate each option:

(a) (p ∧ ¬p): This is a **contradiction** because **p** and **¬p** cannot both be true at the same time. The conjunction (AND) of a statement and its negation will always be **false**.

· **Example:** If p = true, ¬p = false and vice versa.

(b) (p ∨ ¬p): This is a **tautology** because **p** or **¬p** must always be true. One of the two will always be true for any value of **p**.

· **Example:** If p = true, then (p ∨ ¬p) is true; if p = false, then (p ∨ ¬p) is also true because ¬p will be true.

(c) (p → q) ∧ (¬q → ¬p): This is a **logically equivalent** statement but not necessarily a tautology. It represents **the contrapositive** of an implication, and in some cases, it could be false. For example, if **p** is true and **q** is false, then both implications may fail, making the entire conjunction false.

(d) (¬p ∨ p) ∧ (q → q):

· The first part, **(¬p ∨ p)**, is a **tautology** (since one of them must be true).

· The second part, **(q → q)**, is also a tautology (an implication of the same statement is always true).

· Therefore, the entire expression is a tautology. This is valid, but **(b)** is simpler and always true.

Information Booster:

1. **Tautology**: A logical formula that is always true for all possible truth values of the variables.
2. **Contradiction**: A logical formula that is always false, no matter the truth values.
3. **Contingency**: A formula that is neither a tautology nor a contradiction — it can be true in some cases and false in others.

Q2. A CPU h[redacted]E), Memory (M) and Wr[redacted] no pipeline stalls and th[redacted]e execution of 10 instructions?

1. 10
2. 14
3. 15
4. 19

Answer:
B
Sol:
In a 5-stage pipeline, the number of cycles to complete n instructions can be calculated as:
1. Initial Pipeline Filling: Requires 5 cycles (one for each stage).
2. Remaining Instructions: Each new instruction completes in 1 additional cycle.
Total Cycles = 5 + (10 − 1) = 5 + 9 = 14

Information Booster:
Pipeline Stages:
Fetch (F): Retrieves instruction from memory.
Decode (D): Decodes the instruction.
Execute (E): Executes the instruction.
Memory (M): Accesses memory if needed.
Write-back (W): Writes results back to register.
Additional Knowledge:
Without any stalls, pipelines increase instruction throughput but not necessarily the execution speed of individual instructions.

Q.3 A CPU uses scaled-indexed addressing where:
EA = Base + (Index × Scale)
If Base = 4500H, Index register contains 3AH and Scale = 8, what is the effective address?

A. 452EH
B. 46D0H
C. 4688H
D. 4518H

Answer: B

Sol: The Effective Address (EA) is calculated using the formula:

$$\text{EA} = \text{Base} + (\text{Index} \times \text{Scale})$$

1. **Identify values:**
- $\text{Base} = 4500\text{H}$
- $\text{Index} = 3\text{AH}(\text{Decimal}58)$
- $\text{Scale} = 8$

2. Calculate the Offset:

$$\text{Offset} = 3\text{AH} \times 8 = 58_{\text{dec}} \times 8_{\text{dec}} = 464_{\text{dec}}$$

3. Convert Offset to Hexadecimal:

$$\text{Offset} = 464_{\text{dec}} = 1\text{D0H}$$

4. Calculate Effective Address (Actual Calculation):

$$\text{EA} = 4500\text{H} + 1\text{D0H} = 46\text{D0H}$$

Information Booster

Scaled-indexed addressing is a highly efficient addressing mode used frequently in modern processors, particularly for accessing elements in arrays.

1. **Purpose of Scaled-Indexed Addressing:**

· This mode is essential for programming high-level languages like C/C++, particularly when dealing with **arrays and structures**.
· It allows accessing the i-th element of an array with a single memory access instruction, as the scaling factor automatically handles the size of the data type (e.g., 1 for bytes, 4 for 32-bit integers, 8 for 64-bit floating point numbers).
2. **Components:**
· **Base Register:** Holds the starting memory address of the data structure (e.g., the base address of an array).
· **Index Register:** Holds the index or subscript of the element to be accessed (e.g., the array index i).
· **Scale Factor:** A small constant (usually 1, 2, 4, 8) representing the size (in bytes) of the data element.
3. **Mechanism:** The CPU performs the entire calculation ($\text{Index} \times \text{Scale}$) and the final addition in the Address Generation Unit during the instruction execution phase, without needing separate arithmetic instructions.

Additional Knowledge

· **Addressing Modes:** Define how the operand of an instruction is located. Other common modes include:
· **Immediate:** The operand value is directly in the instruction.
· **Register:** The operand is in a CPU register.
· **Direct:** The effective address is specified directly in the instruction.
· **Register Indirect:** The address of the operand is in a register.
· **Displacement/Relative:** $\text{EA} = \text{Base} + \text{Displacement}$.
· **Memory Access Time:** The use of scaled-indexed addressing helps in optimizing memory access by reducing the number of instructions required to calculate the final address, thereby improving the overall CPI (Cycles Per Instruction) for array access loops.

Q.4 Match the LIST – I with LIST – II.

	List – I (Boolean Algebra Law)		List – II (Axioms)
A.	Absorption Law	I.	$a + 1 = 1$
B.	Bounded Law	II.	$a + 0 = a$
C.	Identity Law	III.	$a^*(b + c) = (a^*b) + (a^*c)$
D.	Distributive Law	IV.	$a + (a^*b) = a$

Choose the correct answer from the options given below:

Match the columns.

A. A-IV, B-I, C-II, D-III
B. A-IV, B-III, C-I, D-II
C. A-III, B-IV, C-II, D-I
D. A-II, B-III, C-IV, D-I

Answer: A

Sol: We are asked to match the Boolean Algebra Laws with their corresponding axioms.

A. Absorption Law:

• The **Absorption Law** in Boolean algebra states:

$a + (a \cdot b) = a$

• This matches with **IV.** $\mathbf{a + (a \cdot b) = a}$, which is the axiom corresponding to the Absorption Law.

B. Bounded Law:

• The **Bounded Law** in Boolean algebra states:

$a + 0 = a \quad \text{and} \quad a \cdot 1 = a$

• This matches with **I. a + 0 = a**, which is the axiom corresponding to the Bounded Law.

C. Identity Law:

• The **Identity Law** in Boolean algebra states:

$a + 1 = 1 \quad \text{and} \quad a \cdot 0 = 0$

• This matches with **II. a + 1 = 1**, which is the axiom corresponding to the Identity Law.

D. Distributive Law:

• The **Distributive Law** in Boolean algebra states:

$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

• This matches with **III.** $\mathbf{a \cdot (b + c) = (a \cdot b) + (a \cdot c)}$, which is the axiom corresponding to the Distributive Law.

Information Booster:

1. Absorption Law allows simplification of Boolean expressions by eliminating redundant terms.

2. Bounded Law establishes the identity elements for the operations of addition (0) and multiplication (1).

3. Identity Law ensures that certain operations (adding 1 or multiplying by 0) have a consistent result.

4. Distributive Law allows us to "distribute" one operation over another, making it easier to simplify complex Boolean expressions.

Q5. In 8085 [redacted]rrupt Service Routin[redacted]t is at which location?

1. 3CH
2. 24H
3. 74H
4. 34H

Answer:

B

Sol:

In the 8085 [redacted]upt. It is the highest prio[redacted] transferred to a predefi[redacted]

The fixed ISR location for TRAP is:

So, when a TRAP interrupt occurs, the processor jumps to memory location 0024H to execute the corresponding Interrupt Service Routine (ISR).

Information Booster:

1. The TRAP interrupt has the highest priority and cannot be masked or ignored.
2. It is vectored to a fixed address: 0024H in memory.
3. It is used for critical or emergency interrupts, such as power failure or emergency shutdown signals.

Additional Knowledge:

3CH: This is the ISR address for the RST 7.0 interrupt.

34H: This is the ISR address for the RST 6.5 interrupt.

74H: This is not a valid interrupt vector in 8085. It is a confused value, not associated with any defined ISR.

Q6. In the basic computer's interrupt cycle, when an interrupt is acknowledged, the Program Counter (PC) is saved in memory location 0, and the control jumps to address 1 to begin the service routine. Which control signal or condition causes the jump to address 1?

1. IEN = 1 and interrupt flag = 1
2. SC = 0 and AC ≠ 0
3. R = 1 and T_0 = 1
4. PC overflow during fetch

Answer:

A

Sol:

In the basic contents of the Program mber where to return af xecuting the interrupt se

The jump to tro Enable (IEN) signal and t if the t. When IEN = 1 (interru 1 (indic cessor saves the PC and t ervic rout Thus, the co nd i rupt ress 1 after storing the I

Correct answ lag = 1

Information

IEN (Interru t enabl ables t

Interrupt fl or internal condition.

When both e, the acknowledges the int saves t PC jump he interrupt service rout dr s 1

This mecha sures the CPU espon to int up only wh abl d u , preventing unwanted i nti

Additional K

1. Option (b uencing and accumulato

2. Option (c y cause the interrupt jump to address 1.

3. Option (d) PC overflow during fetch: Incorrect; PC overflow is unrelated to interrupt acknowledgment and jump.

Q7. Techniques for performing I/O:

A. Programmed I/O

B. Interrupt-driven I/O

C. Rotational I/O

D. Direct memory access

E. Channelized I/O

Choose the correct answer from the options given below:

1. A and C Only

2. A, B and D Only
3. A, D and E Only
4. B and E Only

Answer:
B

Sol:
Programmed I/O, Interrupt-driven I/O and Direct Memory Access (DMA) are techniques used for performing I/O operations. Programmed I/O involves the CPU actively checking I/O status. Interrupt-driven I/O allows the CPU to perform other tasks until an I/O operation completes and generates an interrupt. DMA enables direct data transfer between memory and I/O devices without continuous CPU involvement, freeing the CPU for other tasks.

Q8. Which i
1. Save PC
2. Fetch IS
3. Resume
4. Save PC

Answer:
A

Sol:
In the conte sor must ng the state of the curre the xt in he Program Counter (PC ng an rrupt eps:
1. Save PC: ction to ex ute (so that the processor ca on in the program a
2. Fetch ISR he Inter Service R s address is typically sto
3. Jump to ro n t handle the interrupt).
4. Resume : After e finis s e ng e proc ssor w ur to point in the program wh was interr pt (us ng e sav PC lue).
Step-by-Ste ked of O tions:
(a) Save PC
This is the c e processor jumps to th here it was interrupted
This is the correct answer.
(b) Fetch ISR address → Save PC → Jump to ISR → Resume program:
This sequence is incorrect because the PC should be saved before jumping to the ISR, not after fetching the ISR address. The program counter must be saved before the interrupt handling begins to preserve the state.
(c) Resume program → Save PC → Fetch ISR address → Jump to ISR:
This sequence is incorrect because resuming the program would occur after the interrupt handling is done, not before the interrupt cycle begins.
(d) Save PC → Resume program → Jump to ISR → Fetch ISR address:
This sequence is incorrect because resuming the program should happen after the interrupt handling is complete, not before fetching the ISR address.

Conclusion:
The correct sequence of actions during an interrupt cycle is:
(a) Save PC → Fetch ISR address → Jump to ISR → Resume program
Information Booster
Interrupt Handling: The interrupt handling process ensures that when an interrupt occurs, the processor can pause its current task, handle the interrupt, and then return to its previous task without losing any state.
ISR (Interrupt Service Routine): The ISR is the code that runs when an interrupt is triggered. It is typically designed to handle events such as I/O operations, timers, or external signals.

Q9. Which of the following is not typically true about RISC architectures?
1. Uses large number of registers.
2. Uses sin
3. Has a lar
4. Uses loa

Answer:
C
Sol:
RISC (Reduc ecture eed in mind.
They typical ructions cle, and rely
heavily on a emory ac rchitecture,
meaning tha s acce mem erformed on
registers. Th ction ith va ISC, because
this describ et Computer). RIS tion set with
fixed instruc
Information
1. RISC Philo
Simplify ins o e ke uic a sing cy e).
Emphasis o re si city and efficien y.
2. Registers:
RISC proces ave a large number of gener l-pur po e register ed c me y ations.
Example: M
3. Instructi n, enabling
efficient pip
4. Load-Sto
Only load (LW) and store (SW) instructions access memory.
All arithmetic/logical instructions operate between registers.
5. Instruction Format: RISC instructions are usually fixed-length (like 32 bits) and follow a few formats, which simplifies decoding.
6. Examples of RISC Architectures: MIPS, SPARC, ARM and RISC-V.
7. Advantage of RISC: Better performance through pipelining and compiler optimization.
Additional Knowledge:
Option (a) Uses large number of registers: True for RISC — helps reduce memory access and speed up execution.
Option (b) Uses single-cycle execution for most instructions: True — most RISC instructions are designed to complete in one clock cycle.

Option (c) Has a large instruction set with variable formats: Incorrect — this is a CISC feature. RISC has small instruction set with fixed formats.
Option (d) Uses load-store architecture: True — fundamental principle of RISC designs.
RISC → Small instruction set, fixed format, load-store, large registers.
CISC → Large instruction set, variable format, memory-to-memory operations.

Q10. Assertion: CISC processors are designed to execute a wide variety of complex instructions. Reason: RISC processors use simpler instructions that require more lines of code to achieve complex tasks.
Choose the correct answer:
1. Both Assertion and Reason are correct and Reason is the correct explanation of Assertion.
2. Both Assertion and Reason are correct but Reason is not the correct explanation of Assertion.
3. Assertion is correct but Reason is incorrect.
4. Both Ass [redacted]

Answer:
B
Sol:
Assertion is [redacted] rm multiple
operations i [redacted]
Reason is c [redacted] explain the
assertion ac [redacted] ex tasks, but
this does no [redacted]
Information [redacted]
1. CISC archi [redacted] rm multiple
operations, [redacted], in a single
instruction. [redacted]
2. RISC archi [redacted] ng a smaller
number of i [redacted]
3. The stater [redacted] rue, but this
is balanced [redacted]
4. CISC can [redacted] e more CPU
cycles per in [redacted]
Additional K [redacted]
RISC proces [redacted] ir efficiency
and simplici [redacted]
CISC proces [redacted] older Intel-
based archit [redacted]

Q11. Consider a 5-stage RISC pipeline: IF, ID, EX, MEM, WB. An instruction sequence is:
I1: LOAD R1, 0(R2)
I2: ADD R3, R1, R4
I3: SUB R5, R3, R6
If there is no forwarding and pipeline stalls are inserted to resolve data hazards, how many stall cycles are required?
1. 1
2. 2
3. 3
4. 4

Answer:

D
Sol:
With no forwarding, a consumer must wait until the producer writes back its result, and only then can the consumer read it in ID.
For I1 (LOAD) → I2 (ADD): The loaded value of R1 becomes available only in I1.WB (cycle 5 if I1 starts at cycle 1). Therefore, I2 must delay its ID from cycle 3 to cycle 5, inserting 2 stalls (the classic load-use penalty without forwarding).
For I2 (ADD) → I3 (SUB): With no forwarding, R3 is usable only after I2.WB (cycle 8). After the first hazard's delay, I3 would reach ID at cycle 6; it must be postponed to cycle 8, adding 2 more stalls.
Total stalls = 2 + 2 = 4
Information Booster:
1. No-forwarding rule of thumb: A dependent instruction can safely read a register only when the producer ha
2. Typical pe
LOAD → ALU
ALU → ALU
the common
read-after-w
3. Pipeline i
the pipe as
4. Compiler
consumer ca
5. Register f
can reduce
conservative
6. Why load-
wait until W
7. Peak IPC
degrade CPI

Q12. Arrang
A. Initiate D
B. Transfer
C. Processor
D. DMA cont
Choose the
1. C, A, B, D
2. A, C, B, D
3. A, B, C, D
4. C, B, A, D
Answer:
B
Sol:
The steps for DMA transfer:
1. Initiate DMA transfer request.
2. Processor grants DMA control over the system bus.
3. Transfer data directly between peripheral and memory.
4. DMA controller completes data transfer and signals completion.
Information Booster:

1. DMA (Direct Memory Access) allows peripherals to communicate directly with memory, bypassing the CPU for faster transfers.
2. Processor Grant: The CPU grants control to the DMA controller, freeing up the CPU to perform other tasks while the transfer takes place.
3. Transfer Process: Data is transferred directly between the peripheral and memory, improving efficiency.
Additional Knowledge:
DMA Controllers are particularly important in high-performance systems where the CPU needs to be freed from managing routine I/O tasks.
Interrupt: Once the transfer is complete, the DMA controller sends an interrupt to signal completion.

Q13. Which of the following best describes the advantage of using DMA over programmed I/O?
1. It allows CPU to supervise every data transfer cycle
2. It increa
3. It bypass
4. It avoids

Answer:
B
Sol:
DMA increa he CP ess and data transfer. Ins r the res I/O devices and memor ws it to pe erall system efficiency.
Information
1. DMA Ove (DMA) is a system O devices to transfer dat y without involving
2. Programm med U sup between the I/O devices ferring data directly, fre U h o ask
3. CPU Utili MA nces C utiliz tion by CPU spends managing da sfe al vin t pe orm other t s duri those e ds
4. Efficiency A: DMA in rea s syst efici y y reduci ov r ad t U, especially in data-inte
Additional K
Incorrect Op cause one of the primary advantages of DMA is th to supervise every data transfer cycle. In programmed I/O, the CPU handles every transfer, but DMA allows the I/O devices to communicate directly with memory, reducing CPU involvement.
Incorrect Option (c): "It bypasses memory to directly access I/O devices" is incorrect because DMA does not bypass memory; it uses memory as an intermediate storage location for data. DMA does not bypass memory. Instead, it allows I/O devices to transfer data directly to and from memory without CPU intervention. DMA uses memory as a staging area for data, so this statement is not accurate.
Incorrect Option (d): "It avoids use of buses by using registers for communication" is incorrect because DMA still uses buses to transfer data between I/O devices and memory. DMA still uses system buses to transfer data between memory and I/O devices. While DMA helps to reduce CPU interaction, it does not avoid the use of buses; rather, it uses the buses more efficiently, without involving the CPU in each step.
Read the given passage and answer the following questions.

In computer system architecture, address sequencing refers to the method by which memory addresses are generated and accessed during the execution of a program. The process of generating memory addresses is crucial for the efficient retrieval and storage of data in memory. Address sequencing can be implemented in several ways, depending on the system's design, such as linear, interleaved, or segmented addressing.

In linear addressing, a simple, continuous range of memory addresses is used. This means that memory locations are arranged sequentially, and the CPU accesses them in a straight line, one after another. This type of addressing is commonly used in systems where the data structure is contiguous, such as arrays or linear lists.

Interleaved addressing divides the memory into several banks, each of which is accessed independently. This allows for faster memory access as different parts of the memory can be accessed simultaneously, improving the system's overall performance, especially in parallel processing tasks. The primary advantage of interleaving is that it reduces access time by balancing the load across multiple me

On the othe ments, each containing a are typically used in syst . Segmented addressing ent can be independen

The choice efficie ter system, influencing peed, p mplexity of memory ma electi g th g method is fundamenta com r sys

Q14. What i sequencing in co

1. To optim ecution.
2. To gener resse
3. To mana
4. To desig in t s s.

Answer:

B

Sol:

The main p e and access memory ad d by which memory ad crucial for retrieving a the overall performanc

Analysis of the options:

Option (a): While address sequencing can indirectly impact processor execution speed, its primary goal is not specifically to optimize processor execution but to handle memory address generation and access efficiently.

Option (b): Correct. The main focus of address sequencing is to ensure that memory addresses are accessed in a way that enhances the system's efficiency, particularly in terms of memory access speed and storage retrieval.

Option (c): Address sequencing is not focused on managing the CPU's cache memory, though efficient memory access can impact cache performance.

Option (d): Address sequencing is not directly related to designing input/output systems; it is more focused on memory management and address generation.

Q15.

Which addressing scheme divides the memory into multiple banks, improving access time by allowing simultaneous access?

1. Linear addressing
2. Segmented addressing
3. Interleaved addressing
4. Direct addressing

Answer:

C

Sol:

Interleaved addressing is a memory access technique that divides the memory into multiple banks. Each bank can be accessed independently, allowing simultaneous access to different parts of the memory. This techni [redacted] ystems with parallel pro [redacted] ces the load across mult [redacted] emory. This method enh [redacted] ry retrieval compared to [redacted]

Information [redacted]

1. Types of [redacted]

Linear Addr [redacted] and the CPU accesses the [redacted] not provide performanc [redacted]

Segmented [redacted] articular set of instructio [redacted] ove memory access speed [redacted]

Interleaved [redacted] be accessed simultaneou [redacted] to work in parallel, ma [redacted]

Direct Addr [redacted] hile simple, it does not p [redacted]

2. Applicatio [redacted]

Parallel Pro [redacted] rocessors or threads nee [redacted] multi-core systems.

High-perfor [redacted] tions, large-scale data p [redacted] is critical.

3. Advantag [redacted]

Improved Memory Access Time: By distributing memory accesses across multiple banks, interleaved addressing reduces memory contention and latency.

Better Utilization of System Resources: It maximizes the throughput of memory systems, especially in systems that require frequent and parallel memory accesses.

Scalability: As the number of memory banks increases, the system can handle more data access requests simultaneously, making it scalable for high-demand applications.

4. Disadvantages of Interleaved Addressing:

Complexity in Memory Management: Dividing memory into banks requires careful management to ensure that data is distributed optimally across the memory modules.

Cost and Hardware Requirements: Implementing interleaved addressing typically requires additional hardware, which may increase the overall cost of the system.

Additional Knowledge:

Option (a) - Linear addressing: Linear addressing does not involve dividing memory into multiple banks. It simply arranges memory in a continuous sequence, which may be effective for simple data structures, but it doesn't support parallel access or performance optimization like interleaved addressing.
Option (b) - Segmented addressing: While segmented addressing divides memory into segments for better organization and protection, it does not inherently improve access time or support parallel memory access. It is more useful in systems that require logical separation of memory.
Option (d) - Direct addressing: Direct addressing directly refers to memory access using a specific memory address. This method doesn't involve optimizing memory access times or enabling simultaneous access across multiple memory banks.

Q16. What is the primary advantage of interleaved addressing?
1. It allows
2. It reduce
3. It protec
4. It makes

Answer:
B
Sol:
The primary ng is t ing the load across mult e memo addressing allows simu ons of mer up memory retrieval, es requ and c uch as those used in para ocess stem formance by preventing l cessing memory s
Information
1. What is In
Interleaved ccessing the memory se (o s and e system can access differ mory banks simultaneou ving ple da items to be t in pa lle si ly improves access time mo th ug ut.
2. Applicatio terleaved ad ssing:
Parallel Com g ed addr ssing is especia y useful in parallel compute tems, where multiple pr mory access efficiency ar
High-Perfor in achieving faster data a systems.
3. Advantages:
Reduced Latency: Since memory can be accessed from different banks simultaneously, the overall latency of memory retrieval is significantly reduced.
Better Memory Utilization: By distributing the load across multiple memory banks, the system makes better use of available memory bandwidth.
Improved Throughput: In systems with high parallelism, interleaved addressing can dramatically increase the throughput of memory operations.
4. Disadvantages:
Complexity in Implementation: Implementing interleaved addressing requires additional hardware to manage multiple memory banks and coordinate the simultaneous accesses.
Potential for Unbalanced Access: If memory is not evenly distributed, some banks may become more heavily loaded than others, reducing the effectiveness of interleaving.

Additional Knowledge:
Option (a) - Sequential Memory Access: Incorrect. Sequential access is the opposite of what interleaved addressing does. Interleaved addressing enables parallel access across multiple memory banks, not sequential access.
Option (c) - Memory Protection: Incorrect. While memory protection can be important for system stability and security, it is not the primary focus of interleaved addressing. Memory protection is typically handled by other mechanisms, like access control and segmentation.
Option (d) - Complexity in Memory Management: Incorrect. While it's true that interleaving may introduce some complexity in memory management (due to the need for managing multiple banks), the primary advantage of interleaved addressing is the reduction in access time and the ability to improve performance through parallel access, which outweighs this complexity.

Q17. Which addressing scheme is commonly used in systems with data structures that are contiguous, such as arra
1. Linear a
2. Segment
3. Interleav
4. Indexed
Answer:
A
Sol:
Linear addr ms w ere d ch as arrays.
In linear ad arra d in a nd the CPU
accesses th r an This effective for
accessing da ctures because it a to memory
locations. A ous data structures addressing,
as it maint in m thout or memory
managemen
Information
1. Linear Ad
Definition: L dr ssi ref s to as nin memory addres s sequ ly i a uous range.
This makes ghtforwar to ce s c tigu me ry regi
Use Case: Ar exam e of da struc res that benefit from linear a sing, as their
elements ar the starting
address.
Advantages: mplement.
Disadvantag fragmented
or non-contiguous memory.
2. Segmented Addressing:
Definition: In segmented addressing, memory is divided into segments (such as data, code, or stack segments), each with a distinct starting address.
Use Case: Often used in systems that need logical separation of memory areas but not efficient for contiguous data structures.
Disadvantages: Can lead to fragmentation and inefficiencies when accessing large, contiguous data structures.
3. Interleaved Addressing:
Definition: Memory is divided into multiple banks that can be accessed simultaneously, reducing access time.

Use Case: Useful in systems with parallel processing, not specifically for contiguous data structures like arrays.
4. Indexed Addressing:
Definition: This method uses an index to access memory locations, often through an index register.
Use Case: Commonly used in array access but not the primary addressing method for contiguous memory; it's a form of addressing rather than a method for managing entire memory spaces.
Additional Knowledge:
Linear Addressing: This is the most basic and straightforward addressing scheme, suitable for simple data structures like arrays where the memory layout is contiguous. This scheme is also highly efficient in terms of both implementation and access time.
Array Access: In languages like C, an array is essentially a pointer to the start of a contiguous block of memory. This is why linear addressing works well for arrays: accessing the i-th element of an array is equivalent t

Q18. What i
1. It enable
2. It reduce
3. It allows
4. It simpli
Answer:
C
Sol:
Segmented a nct purpose, such as a c segmented addressing i rent parts of the program and allocate memory bas protected or allocated, er.
Information
1. Logical Pa g:
Definition: S be logically organized, s n the nature and purpose
Use Case: O anagement. Each segme erent access rights for co
2. Memory P
Definition: By separating different areas of memory into segments, systems can implement protections like preventing code from accidentally modifying data or preventing programs from accessing memory that they shouldn't. This helps in ensuring the stability and security of the system.
Example: In early Intel x86 architecture, segments like CS (Code Segment), DS (Data Segment), and SS (Stack Segment) were used to differentiate between different kinds of memory, allowing easier protection mechanisms.
3. Advantages of Segmented Addressing:
Flexibility: Allows different-sized segments for different needs (e.g., stack size and data size).
Protection: Memory protection mechanisms can be easily implemented by isolating different memory regions.
Efficient Memory Management: Helps in managing large programs by dividing them into manageable sections, improving both the program structure and memory allocation.

4. Disadvantages: Fragmentation: While segmentation helps in logical partitioning, it can also lead to external fragmentation. If segments are not efficiently sized, free memory between them might become fragmented, making it harder to allocate memory.
Additional Knowledge:
Option (a) - It enables faster parallel processing: Incorrect. Segmented addressing doesn't specifically enhance parallel processing, which is more influenced by methods like interleaved addressing or multi-threading. Segmentation is more about logical memory division and protection.
Option (b) - It reduces memory fragmentation: Incorrect. While segmentation can organize memory, it can actually increase fragmentation (external fragmentation) because memory blocks (segments) can become scattered over time, especially if memory is allocated and freed dynamically.
Option (d) - It simplifies memory management: Incorrect. While segmentation provides organizational benefits, managing segments can be more complex compared to simpler memory models like linear addressing, [redacted]

Q19. Which [redacted] s of Natural
Language P [redacted]
1. Sentime [redacted]
2. Speech [redacted]
3. Packet R [redacted]
4. Machine [redacted]
Answer:
C
Sol:
Natural Lan [redacted] terpret, and
generate h [redacted] on, machine
translation, [redacted] sk that deals
with forwar [redacted]
Information [redacted]
1. Sentimen [redacted]
2. Speech R [redacted]
3. Machine [redacted] te).
4. NLP deals [redacted]
5. Packet ro [redacted]
Additional [redacted]
(a) Sentime [redacted]
(b) Speech [redacted]
(d) Machine [redacted]

Q20. Which of the following statements is the best description of supervised learning?
1. 'If a particular input stimulus is always active when a neuron fires then its weight should be increased.'
2. 'If a stimulus acts repeatedly at the same time as a response, then a connection will form between the neurons involved. Later, the stimulus alone is sufficient to activate the response.'
3. 'The connection strengths of the neurons involved are modified to reduce the error between the desired and actual outputs of the system.'
4. None of the above
Answer:
C
Sol:

Supervised learning is a type of machine learning where the model learns from a labeled dataset, meaning each input is paired with the correct output. The learning process involves adjusting the parameters, such as weights in neural networks, to minimize the difference (error) between the predicted output and the actual desired output. Statement (c) correctly describes this process by stating that the connection strengths of neurons are modified to reduce the error, which aligns perfectly with the core principle of supervised learning — learning through error correction using known outputs. This makes option (c) the best description among the given choices.

Information Booster:

1. Supervised learning uses labeled data to train models by minimizing error.
2. The error is the difference between the predicted output and the actual (desired) output.
3. Weight adjustments are made systematically to reduce this error, commonly using algorithms like gradient descent.
4. This proc e.
5. It contras
6. Examples
7. Neural n cy based on errors.

Additional K

• Option (a) gether") but does not ad learn

• Option (b) ssociati in machine learning cor

• Option (d) ccur y des

Q21. A moc s as "spam" or 'n s. This is an illustration

1. Unsuper
2. Supervis
3. Reinforc ni
4. Transfer

Answer:

B

Sol:

Supervised ut (label) is provided fo ails that are already tag predictions accordingly.

Information Booster:

1. Supervised learning uses input-output pairs to learn a mapping function.
2. It is widely used in classification (e.g., spam detection) and regression (e.g., price prediction).
3. It requires a high-quality labeled dataset for effective model training.

Additional Knowledge:

Unsupervised learning: Learns from unlabeled data to discover hidden patterns or groupings, like in clustering or dimensionality reduction.

Reinforcement learning: Involves an agent that learns by interacting with an environment and receiving rewards or penalties; not based on labeled data.

Transfer learning: Applies a model trained on one task/domain to a related but different task, often with limited data in the target domain.

Q22. Among the various AI learning paradigms, which technique involves learning from unlabeled data and finding hidden patterns or intrinsic structures within the data?

1. Semi-Supervised Learning
2. Self-Supervised Learning
3. Unsupervised Learning
4. Active Learning

Answer:
C
Sol:
Unsupervised Learning allows AI models to discover patterns and relationships in unlabeled data without human intervention. It is widely used in clustering, anomaly detection and association rule mining, making it essential for uncovering hidden insights in vast datasets.
Information
1. Definition find natural patterns or
2. Techniqu (e.g., PCA, t-SNE).
3. Use Cases ession.
Additional K
• Semi-Supe d unl
• Self-Super ion whe r learning.
• Active Lea or la eling

Q23. Consid ach an co a minimum of 2 keys (d need to store 120 e minimum number of n ys while maintaining

1. 30
2. 45
3. 20
4. 18

Answer:
B

Sol: Given:

- **B* Tree properties:**
  - **Order of B* Tree:** Each node can hold a maximum of 4 keys and a minimum of 2 keys.
  - **2/3 full property:** Each node must be at least **2/3 full.**
- **Total number of keys** = 120 keys.

Step 1: Initial Calculation

- **Minimum number of nodes** required (without considering the 2/3 full property):

$$\text{Minimum number of nodes} = \frac{\text{Total keys}}{\text{Minimum keys per node}} = \frac{120}{2} = 60 \text{ nodes}$$

This is the **initial estimation.**

Step 2: Considering the 2/3 Full Property

- Since each node must be at least **2/3 full**, we need to **adjust** the calculation. The average number of keys per node should be roughly 2.67 (as 2/3 of 4 keys).
- **Recalculate the number of nodes** using the **average number of keys per node** (approximately 2.67):

$$\text{Adjusted number of nodes} = \frac{\text{Total keys}}{\text{Average keys per node}} = \frac{120}{2.67} \approx 45 \text{ nodes}$$

The **minimum number of nodes** required to store 120 keys in the B* Tree is **45 nodes**, as the nodes need to be distributed properly according to the **2/3 full property.**

Information Booster:

1. B* Tree Properties:

o A B* Tree is a balanced tree that maintains its properties through the **2/3 full property**, ensuring that each node (except the root) contains at least 2/3 of its maximum capacity.

o Order of B* Tree refers to the **maximum number of children** each node can have, which directly influences the number of keys that can be stored.

2. 2/3 Full Property: The **2/3 full property** ensures that nodes are kept relatively **balanced** in terms of key storage, ensuring efficient searching and maintaining low height in the tree.

3. Efficiency of B* Tree: B* Trees are typically used in databases and file systems where **efficient search**, **insertion** and **deletion operations** are crucial. They are preferred for systems that need to handle large datasets due to their **balanced nature** and ability to store many keys in each node.

4. Height of B* Tree: The **height** of a B* Tree directly impacts the **performance** of search, insertion, and deletion operations. Lower tree heights lead to faster access times because fewer nodes need to be traversed.

5. Applications of B* Trees: B* Trees are commonly used in applications like **databases** and **file systems** where large amounts of data need to be stored and accessed efficiently. Examples include **SQL databases** and **indexing systems.**

Additional Knowledge:

B* Tree vs B - Tree: B* Trees are a variant of B - Trees, with additional refinements such as the 2/3 full property, which ensures that the tree remains **more balanced** compared to regular **B - Trees**, especially under frequent updates.

Q24. Consider the statements regarding generalization and specialization:

1. Generalization is top-down, specialization is bottom-up.
2. Generalization reduces schema size, specialization expands it.
3. Both always produce the same entity sets.
4. Both remove inheritance.

Which statement is true about generalization and specialization?

1. Generalization is top-down, specialization is bottom-up
2. Only 2
3. Both always produce the same entity sets
4. Both remove inheritance

Answer:

B

Sol:
In generalization, multiple lower-level entities with common attributes are combined into a higher-level entity, leading to schema simplification and size reduction.
In specialization, a higher-level entity is divided into multiple lower-level entities (subclasses) based on distinguishing features, which leads to schema expansion.
Hence, the statement that generalization reduces schema size and specialization expands it is true.
Information Booster:
1. Generalization is a bottom-up approach, combining entities to simplify design.
2. Specialization is a top-down approach, splitting entities for more detailed modeling.
3. Generalization minimizes redundancy, while specialization adds granularity.
Additional Knowledge:
Generalization is top-down, specialization is bottom-up: Incorrect — specialization is top-down, generalizati
Both always
Both remov

Q25. Which
1. OLAP de
2. OLAP st
3. OLAP fo
4. OLAP is
Answer:
C
Sol:
(c) OLAP fo
Correct: OL
multi-dimen
for routine
transactiona
Information
OLTP (Onlin
Focus: Supp
Data: Stores
Operations:
transactions
Database De
OLAP (Online Analytical Processing):
Focus: Supports complex queries for data analysis.
Data: Stores historical data for trend analysis, business reporting, and decision-making.
Operations: Focuses on complex queries like aggregations, multi-dimensional analysis, and reporting.
Database Design: Denormalized (e.g., using star or snowflake schemas) to improve query performance for analytical queries.
Additional Knowledge:
(a) OLAP deals with transactional updates, OLTP with analytical queries:
Incorrect: This is the opposite of what actually happens. OLAP is used for analytical queries (such as complex aggregations and data analysis) to make business decisions, while OLTP is focused on transactional updates (i.e., processing routine business operations like sales transactions, order processing, etc.).

(b) OLAP stores current data, OLTP stores historical data:
Incorrect: OLTP stores current transactional data because it handles real-time operations (e.g., inventory management, banking). On the other hand, OLAP systems often store historical data because they are used for analytical purposes over long periods (e.g., trend analysis, performance metrics).
(d) OLAP is normalized, OLTP is de-normalized:
Incorrect: Actually, the opposite is true. OLTP systems are typically normalized to reduce redundancy and improve the efficiency of transactional queries. OLAP systems are often denormalized to improve the speed of read-heavy analytical queries and to support multidimensional analysis (e.g., through star schemas or snowflake schemas).

Q26. Which of the following OLAP operations is most computationally expensive on a large data warehouse cube?

1. Slice
2. Dice
3. Roll-up
4. Drill-dov

Answer:
B
Sol:
In Online A arehot dimensional data. The ba , dice, ro ng different ways to nav
OLAP Opera
1. Slice:
A slice oper given dimension dimension. For example icular year or region Computatio cause ducing rking with a subset of th
2. Dice:
The dice op s a m comple operat on tha s It el ts om multiple dimensions te su be For ex pl select n data f a par r c n on of years, regions, and ct categ es.
Dice is com ic more expe ive th slice because it reduces the dat ss multiple dimensions
3. Roll-up:
Roll-up is ar hy in one or more dimen vel.
It is computationally intensive, but less so than dice because it involves aggregation, not the selection of data from multiple dimensions.
4. Drill-down:
Drill-down involves going deeper into the hierarchy, typically to a more granular level (e.g., from yearly data to monthly or from country to city-level data).
This operation involves expanding data rather than aggregating it, which can be computationally intensive, but is generally less costly than dice.
Why Dice is the Most Expensive?
The dice operation selects a subset of data across multiple dimensions. The more dimensions you select, the more computationally expensive the operation becomes because it requires scanning and retrieving specific data from several axes of the cube.

The complexity of the dice operation arises because it must handle multiple filter conditions across multiple dimensions simultaneously.

Information Booster:

1. OLAP Cubes: These are multi-dimensional data structures that allow fast querying and analysis of large datasets. They are designed to be used in data warehousing environments to perform complex queries in an optimized manner.

2. Multi-dimensional Analysis: OLAP operations help users to "slice" and "dice" data across various dimensions to get insights. Dimensions typically represent business hierarchies such as time (year, quarter, month), geography (country, region, city) and other business-related attributes.

3. Data Aggregation: Operations like roll-up and drill-down are primarily used for data summarization or refinement. Roll-up aggregates data to a higher level, while drill-down breaks down data into finer granularity.

Additional K

Efficiency o warehouse design, espe ed for faster query perfo ering across multiple din

Optimizatio e indexing, partitioning sed

Q27. Match t functio

	List – I (Opera	
A.	Slice	orien
B.	Dice	be on multiple di
C.	Roll-up	o higher level
D.	Pivot	cube mens

1. A-IV, B-I
2. A-II, B-IV
3. A-IV, B-I I
4. A-I, B-II, III

Answer:

A

Sol:

1. Slice (A):

Definition: T fic value for one of the d e slice.

Functionality: This corresponds to selecting a subset of the cube on a single dimension.

Match: A → IV

2. Dice (B):

Definition: The Dice operation selects a subset of the cube by specifying values for multiple dimensions. It essentially selects a block (or a slice) of the data.

Functionality: This corresponds to selecting a subset of the cube on multiple dimensions.

Match: B → II

3. Roll-up (C):

Definition: The Roll-up operation aggregates data to a higher level by climbing up the hierarchy in a dimension. For example, you can roll up from days to months or from months to years.

Functionality: This corresponds to aggregating data to a higher level.

Match: C → III

4. Pivot (D):
Definition: The Pivot operation changes the orientation of the data, allowing users to view it from a different perspective (i.e., rotating the cube to swap dimensions).
Functionality: This corresponds to changing dimension orientation.
Match: D → I
Information Booster:
1. Slice: Reduces the dimensionality of the data cube by fixing one dimension.
2. Dice: Allows you to select a specific subset from the data cube by specifying multiple dimensions.
3. Roll-up: Aggregates data, essentially summarizing it at a higher level of the hierarchy (e.g., from individual sales data to yearly totals).
4. Pivot: Rotates the data cube for a different perspective on the data, helping to analyze it from various angles.

Q28. Which ox?
1. They car
2. They car
3. They car
4. They car
Answer:
C
Sol:
Java Applets nt fo secu re restricted in what they vitie
Information
1. Java Appl
A sandbox t isolates untrusted esources to prevent mal
The restrict o the user's system.
2. Same-Ori y: Th ncy en res tha an ap ion y i era h e server that served it, pr g m ic s a let fr a essing ernal s vers.
3. File Syste trictions: pp s a e t a lov d t ead or to lo i em without explicit per ns
4. Security access. For instance, it
5. Network only to the originating s er network-based attacks.
Additional Knowledge:
Option (a) They can read/write local files without restriction → Incorrect, because Java Applets are restricted from reading or writing to the file system to protect user data. Applets are restricted from accessing local files to prevent potential malicious activities. They cannot directly read or write files on the local machine unless they are granted explicit permissions via a security manager (which was rarely the case).
Option (b) They cannot make any network connections → Incorrect, because Applets can make network connections but are restricted to the server from which they were downloaded. Applets can make network connections, but they are restricted to connecting only to the server from which they were downloaded due to the Same-Origin Policy. This prevents them from making arbitrary connections to other servers for security reasons.

Option (d) They can execute operating system commands directly → Incorrect, because Applets are not allowed to execute operating system commands due to security concerns. Applets are restricted from executing operating system commands or accessing system resources for security reasons. This ensures that applets cannot perform malicious operations on the user's system.
Java Applets, once a popular method for creating interactive web applications, are now considered deprecated due to security concerns and the rise of more modern web technologies such as JavaScript, HTML5 and WebAssembly.

Q29. Which of the following is true about the default constructor in Java?
I. A default constructor is explicitly defined by the user.
II. A default constructor takes parameters.
III. A default constructor is automatically provided by Java if no constructors are defined.
IV. A default constructor cannot be overloaded.
1. Only I
2. Only II
3. Only I ar
4. Only III
Answer:
D
Sol:
In Java, a de generate er does not provide any lizes instan s (e.g., 0 for integers, nu
Information
1. A default tically by Java only efined in the class.
2. It does no nitiali les to t
3. If a user te a default constructor.
Example:

```
class Example {
    int value;
    public static void main(String[] args) {
        Example obj = new Example();  // Calls the default constructor
        System.out.println(obj.value);  // Output: 0 (default value of int)
    }
}
```

Additional Knowledge:
A default constructor is explicitly defined by the user: If explicitly defined, it is no longer a default constructor but a no-argument constructor.
A default constructor takes parameters: Default constructors do not take parameters; constructors with parameters are explicitly defined by the user.
A default constructor cannot be overloaded: Constructors can be overloaded by providing different parameter lists.
A default constructor initializes instance variables to custom values: Custom initialization requires an explicitly defined constructor. Default constructors initialize variables to default values only.

Q30. Which of the following are TRUE about constructors in C++?

A. A constructor can be overloaded.
B. A constructor does not have a return type.
C. A constructor must be declared as a friend function.
D. A constructor is called when an object is destroyed.
Choose the correct answer from the options given below:
1. B, C, D only
2. B, C only
3. A, B, C only
4. A, B only
Answer:
D
Sol:
In C++, cons[illegible]ctors do not have return
1. A constru[illegible]
This is true. [illegible]which can be helpful in pr[illegible]
2. A constru[illegible]
This is true. [illegible]do n[illegible]
Information[illegible]
Constructor[illegible]e they ai[illegible]nitialize the object.
Constructor[illegible]crea[illegible]. Wh[illegible]ructors (not constructor[illegible]

Q31. Only le[illegible]
A. pointer +[illegible]
B. pointer -[illegible]
C. pointer +[illegible]po[illegible]
D. pointer -[illegible]poi[illegible]
E. pointer -[illegible]→[illegible]
Choose the [illegible]ppropriate an[illegible]er ro[illegible]he op[illegible]ns gi[illegible]en bel[illegible]
1. A, B, C O[illegible]
2. A, B, D O[illegible]
3. A, B Only
4. A, E Only
Answer:
D
Sol:
Let's analyze each of the given pointer operations to determine which are legal in C programming:
Pointer Operations in C:
1. Pointer + Number → Pointer (A):
Legal: Yes, when you add a number to a pointer, you move the pointer forward by that number of elements (based on the type the pointer points to). For example, if ptr is a pointer to an integer, ptr + 2 will move the pointer ahead by two integers.
2. Pointer - Number → Number (B):
Not Legal: This operation is not legal. Pointer - number is legal, but subtracting a pointer from a number is not. The number should be subtracted from the pointer, not the other way around. So, ptr - 2 (pointer minus number) is legal, but 2 - ptr (number minus pointer) is not valid in C.

In summary:
Correct: Pointer - number is valid.
Incorrect: Number - pointer is invalid.
3. Pointer + Pointer → Pointer (C):
Not Legal: Adding two pointers is not allowed in C. You can only add a number to a pointer, not another pointer. This operation is not valid.
4. Pointer - Pointer → Pointer (D):
Not Legal: Subtracting two pointers results in a number (specifically the difference in the number of elements between them). The result is an integer, not a pointer.
5. Pointer - Pointer → Number (E):
Legal: Yes, subtracting two pointers of the same type gives the difference in the number of elements between them, and the result is an integer (ptrdiff_t).
Correct Con
Legal operat
Final Answe
Information
1. Pointer A for efficient memory ma d:
Valid: Addi racting two pointers (if rray).
Invalid: Add ointer fr
2. Why B is r , as it mean the number of elements nd th point e of the data type the poi
Additional K
ptrdiff_t: Th inters is of type ptrd t represents the differen ddres s of t he type the pointer is po

Q32. Which llowi a corre way t decla a ion l p nt r C?
1. int *func);
2. int (*fun int);
3. int (func
4. int *func

Answer:
B
Sol:
Function pointers allow functions to be passed as arguments, stored in variables, and returned from other functions.
int (*func)(int, int); declares func as a pointer to a function that takes two int arguments and returns an int.
Parentheses around *func are essential to make it a pointer to a function instead of a function returning an integer pointer.
Information Booster:
Use Cases:
1. Function pointers are crucial for implementing callbacks in C.
2. They are used in dynamic dispatch for handling different types of functionalities.
Additional Knowledge:

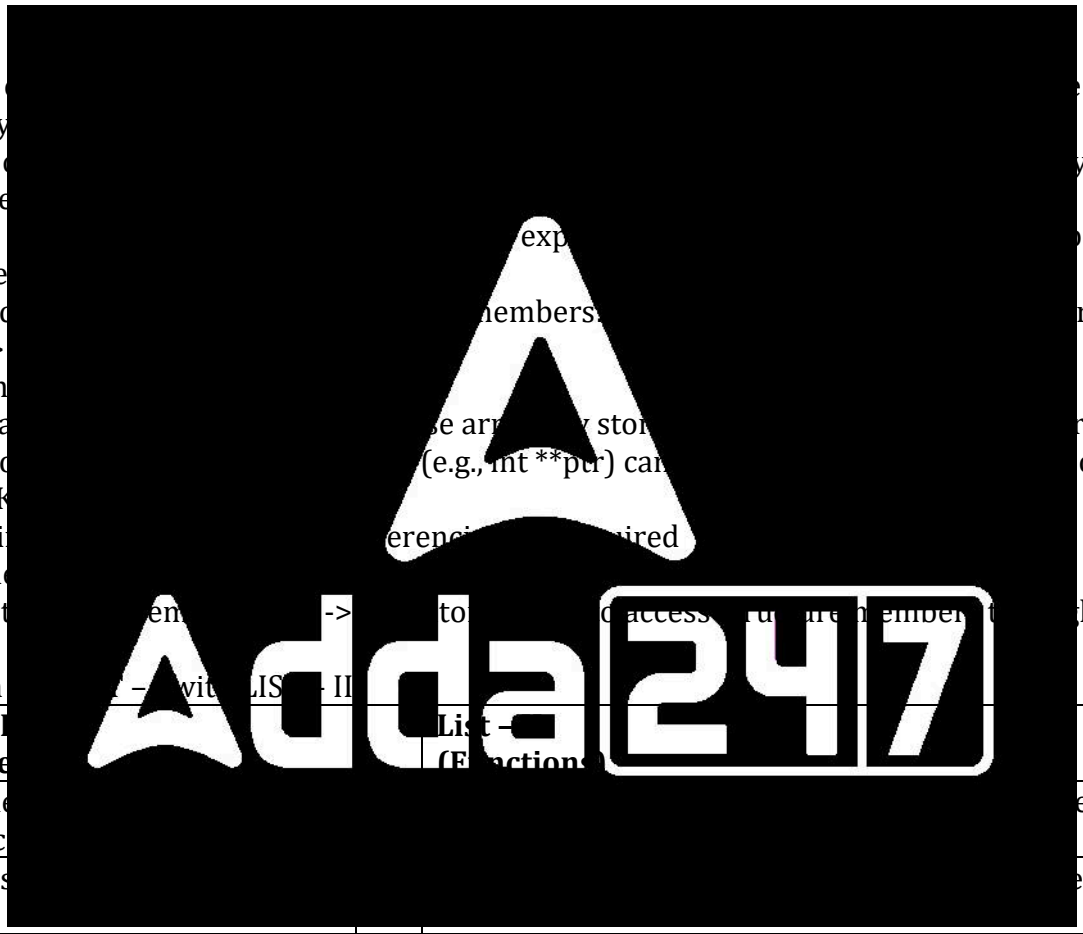
Pointer Syntax Nuances: C's syntax for pointers and function pointers is intricate, often confusing new programmers.
Application in Real-World Programming: Function pointers are heavily used in graphics programming and event-driven programming.

Q33. Given the following array declaration in C:
int arr[5] = {1, 2, 3, 4, 5};
Which of the following pointer expressions refers to the value arr[3]?
1. *(arr + 3)
2. arr[3]
3. *(arr + 4)
4. &arr[3]

Answer:
A
Sol:
In C, arrays r to the first element of t
Let's break
Option (a) *
arr is a poin
arr + 3 mov
* dereferenc
This is the c
Option (b) a
arr[3] direct
This is corre
Option (c) *
arr + 4 mov
* dereferenc
This is incor
Option (d) &
&arr[3] give
This is a refe
This is incor
Conclusion:
The correct
Information

1. Array and Pointer Relationship: In C, the name of an array (arr) is treated as a pointer to its first element. Therefore, expressions like *(arr + n) are valid ways to access array elements, where n is the index.
2. Pointer Arithmetic: Pointer arithmetic allows us to access elements of an array by adding an integer to a pointer. For example, arr + 3 gives a pointer to the fourth element of the array (arr[3]), and dereferencing it (*(arr + 3)) gives the value stored at that location.
3. Dereferencing and Addressing:
Dereferencing a pointer (*ptr) gives us the value stored at the address the pointer is pointing to.
Addressing an element (&arr[n]) gives us the memory address of that element.
Additional Knowledge:
Understanding pointer arithmetic and array indexing in C is crucial for efficient manipulation of data in arrays and memory.

Q34. Which of the following statements about pointers in C are TRUE?
A. Pointers can be used to access array elements.
B. Pointers can store the address of another pointer.
C. Pointers are automatically deferenced in expression.
D. Pointers cannot be used to access structure members.
Choose the correct answer from the options given below:
1. A and C only
2. A and B only
3. B and C only
4. C and D only
Answer:
B
Sol:
A. Pointers (arrays and access array
B. Pointers (y address of other pointe
C. Pointers exp e explicitly dereference
D. Pointers c embers re members using the ->
Information
1. Pointers a e arr z sto rst element.
2. Pointer-to (e.g., int **ptr) ca ointer.
Additional K
Dereferenci erenc ired * operator retrieves th
Accessing St em -> to access h pointers.



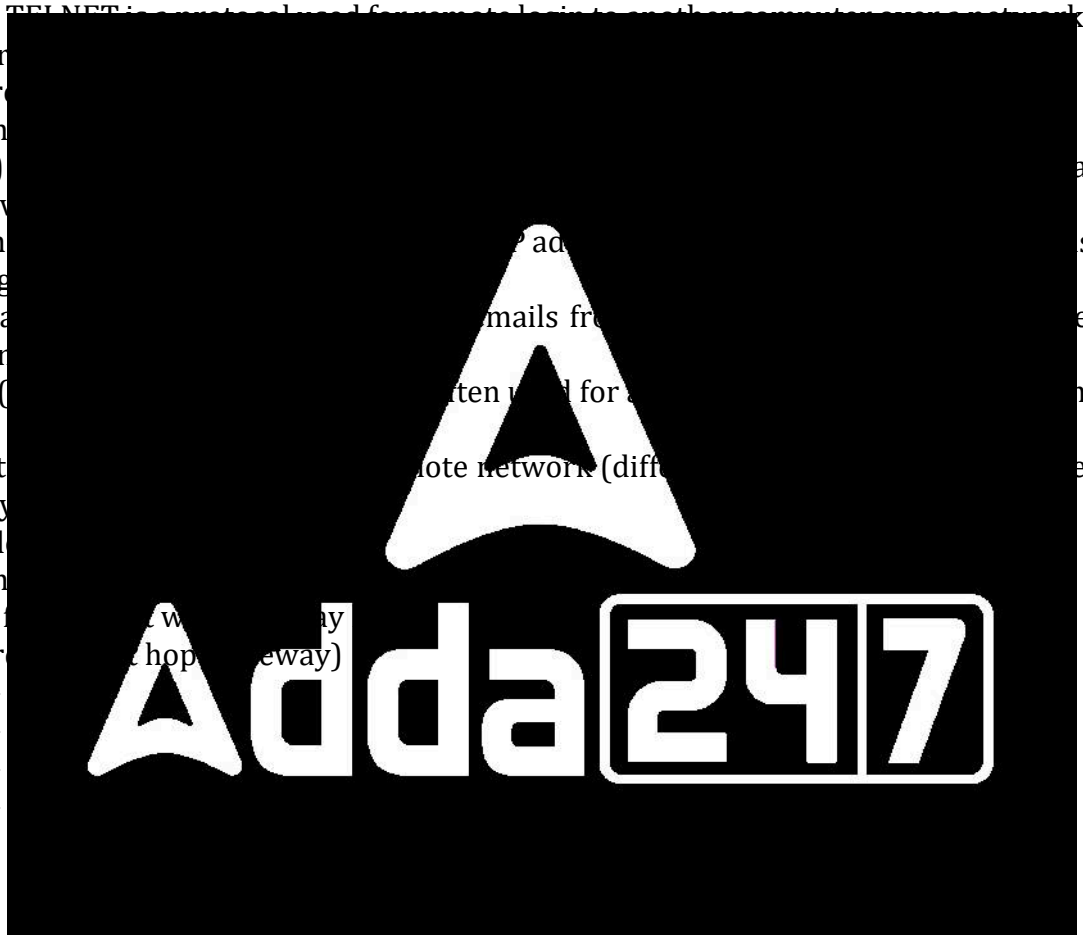
Q35. Match – wit LIS – II

	List – I (Proce		**List – (Functions)**
A.	Interne Protoc		emails
B.	Addres (ARP)		r a
C.	Post Office Protocol (POP)	III.	A protocol that maps IP addresses to MAC addresses on a local area network.
D.	TELNET	IV.	A protocol that allows accessing email on the server without downloading them.

Choose the correct answer from the options given below:
Match the columns.
1. A-II, B-I, C-III, D-IV
2. A-III, B-IV, C-I, D-II
3. A-IV, B-III, C-I, D-II
4. A-IV, B-III, C-II, D-I
Answer:
D

Sol:
Let's break down each process in LIST-I and match it with the correct function from LIST-II:
A. Internet Message Access Protocol (IMAP): IMAP is a protocol used for email retrieval. It allows email clients to access and retrieve emails from a mail server without having to download them. Therefore, IMAP corresponds to IV in LIST-II:
IV. A protocol that allows accessing email on the server without downloading them.
B. Address Resolution Protocol (ARP): ARP is used to map IP addresses to MAC addresses on a local area network. Therefore, ARP corresponds to III in LIST-II:
III. A protocol that maps IP addresses to MAC addresses on a local area network.
C. Post Office Protocol (POP): POP is a protocol used for email retrieval. It allows email clients to download emails from a mail server. Therefore, POP corresponds to II in LIST-II:
II. A protocol that allows email clients to retrieve emails from a mail server.
D. TELNET: TELNET is a protocol used for remote login to another computer over a network. Therefore, TELNET cor
I. Used for r
Information
1. IMAP (A) ation across multiple dev
2. ARP (B) h ad sure proper data routing
3. POP (C) a mails fr ed, they are typically rer
4. TELNET (ten u d for machines.

Q36. A host ote network (diff events (ARP cache empty
1. ARP for d
2. TCP segm
3. Ethernet w ay
4. IP forwar hop eway)
1. 2 → 1 →
2. 1 → 2 →
3. 2 → 3 →
4. 4 → 1 →



Answer:
B
Sol:
When a host sends a TCP segment to a remote network (on a different subnet), and the ARP cache is empty, the following sequence of events occurs:
1. Step 1: ARP for default gateway's MAC:
The host needs to send the packet to a remote network, so it must forward the packet to its default gateway (router). Since the ARP cache is empty, the host will first send an ARP request to resolve the MAC address of the default gateway.
2. Step 2: TCP segment + IP packet built:
After receiving the gateway's MAC address (via ARP reply), the host proceeds to construct the TCP segment (Layer 4) and encapsulate it in an IP packet (Layer 3). The IP packet contains the destination IP address, which is the IP address of the remote network.
3. Step 3: Ethernet frame built with gateway MAC:

The host then builds an Ethernet frame (Layer 2) to deliver the IP packet to the default gateway. The destination MAC address in the Ethernet frame is set to the MAC address of the gateway obtained from the ARP reply.
4. Step 4: IP forwards to next hop (gateway):
The Ethernet frame is transmitted to the default gateway, which will then forward the IP packet to the next hop (either directly or through other routing devices) towards the final destination on the remote network.
Information Booster:
1. ARP (Address Resolution Protocol): ARP is used to map an IP address to a MAC address. If the MAC address is not found in the ARP cache, the host sends an ARP request to the network to retrieve it.
2. Ethernet Frame: The Ethernet frame is the lowest layer in the network stack, which encapsulates the IP packet (Layer 3) and is responsible for transferring the data on the local network using MAC addresses.
3. Routing: T te network.
The host for e packet to
the correct
Additional K
ARP Cache: id repeated
ARP lookup P is used to
resolve the
Default Gate outer tha devices on
other netwo in th sam acket to the
default gate

Q37. What i RP (Address Reso
1. To find t ng to a known IP add
2. To map es.
3. To conve
4. To resol MA es ad
Answer:
A
Sol: The Ad esolution Pro col (A) is us d to f d the M ddre as a ith a known
IP address. T tes wi in a loc l netwo k to match the IP address to ical address
(MAC addre
Information
1. Network ring that IP
addresses a
2. Local Network Operation: ARP functions within the local network, as MAC addresses are not used across multiple networks.
3. ARP Table: Devices maintain an ARP table that stores recent IP-MAC address mappings, which reduces repeated ARP requests.
Additional Knowledge:
Mapping IP to Domain Names: This function is handled by DNS (Domain Name System), not ARP.
NetBIOS Name Resolution: NetBIOS over TCP/IP uses a different process for resolving names to IP addresses.
IP to MAC Resolution: ARP does not work in the opposite direction (MAC to IP); it specifically finds MAC addresses for given IP addresses.
MAC to IP Resolution: Reverse ARP (RARP) was used for this purpose historically but is now largely obsolete.

Q38. A classless address is given as 167.199.170.82/27. The number of addresses in the network is:
1. 64 addresses
2. 32 addresses
3. 28 addresses
4. 30 addresses

Answer:
B
Sol:
The subnet mask of /27 indicates that 27 bits are reserved for the network, leaving 5 bits for host addresses. This provides a total of 25 = 32 addresses.
Information Booster
1. Subnet M
2. Host Calc
Number of h
Total addres
3. Usable Ad g 30 usable addresses.
4. Applicatio g.
Additional K
A /28 subne
Classful add d sub et si
CIDR (Class s va le-le

Q39. In IPv dress 192.168.16. ask and the number of u
1. 255.255.
2. 255.255.
3. 255.255. 4
4. 255.255. 30 h

Answer:
C
Sol: Subnet M This subnet mask leaves s for hosts).
Number of U 6. However, the first add ast address, leaving 14 u
Information Booster:
1. CIDR Notation (/28): The /28 indicates that 28 bits are used for the network portion of the address, and the remaining 4 bits are used for hosts.
2. Subnet Mask: The corresponding subnet mask for /28 is 255.255.255.240, which means that 28 of the 32 bits are fixed for the network portion.
3. Usable Hosts: With 4 bits for hosts, there are 24 = 16 total IP addresses, but only 14 usable because 2 are reserved (network and broadcast addresses).
Additional Knowledge:
Class C Address: 192.168.16.45 falls within the Class C IP address range, where the default subnet mask is 255.255.255.0. However, with CIDR /28, the subnet is divided further into smaller segments.
Network Address Calculation: For 192.168.16.45/28, the network address is 192.168.16.32, and the broadcast address is 192.168.16.47.

Q40. When discussing network addressing, which of the following is a Class C private IP address range?
1. 10.0.0.0 - 10.255.255.255
2. 172.16.0.0 - 172.31.255.255
3. 192.168.0.0 - 192.168.255.255
4. 169.254.0.0 - 169.254.255.255

Answer:
C
Sol: The 192.168.0.0 - 192.168.255.255 range is reserved for private networks and is not routable on the public internet. It is commonly used in home, office and enterprise LAN environments.
Information Booster:
1. Private IP Addresses: Used for internal networks, preventing direct exposure to the internet.
2. Network A [redacted] via a public IP address.
3. Other Priv
Additional K
• 10.0.0.0/8
• 172.16.0.0
• 169.254.0. IP A

Q41. You ar to assig evice. If the subnet mas llowi g sta
I. The IP add 10.1 0.0/2
II. The IP ad t add r the
III. The IP ad k address for the s
IV. The IP ad id IP address for the
1. Only I
2. Only II
3. Only III
4. Only IV

Answer:
A
Sol: To dete [redacted] t's ana ze ste by step.
1. Subnet In
The given su
This means
2. Broadcast
For a /24 subnet, the broadcast address is always the last IP in the range.
In this case, the broadcast address is 10.192.0.255.
3. Network Address:
The network address is the first IP in the subnet range.
In this case, the network address is 10.192.0.0.
4. Valid Host IPs:
Valid host IPs are all the addresses between the network address and the broadcast address, excluding both.
For this subnet, the valid host IP range is 10.192.0.1 to 10.192.0.254.
5. Analysis of the IP Address 10.192.0.79:
The IP 10.192.0.79 falls within the valid host range (10.192.0.1 to 10.192.0.254).
Therefore, it is a valid host address in the subnet 10.192.0.0/24.

Information Booster:
A /24 subnet contains 256 total IPs, where the first is the network address, the last is the broadcast address, and the rest are valid host IPs.
Always calculate the subnet range carefully to determine if an IP belongs to the network.
Additional Knowledge:
1. Subnet Mask Notation:
255.255.255.0 = /24 = 256 total IPs.
Subnet masks define how many bits are reserved for the network portion of the address.
2. Broadcast Address:
It is used for communication with all devices in a subnet and is always the last IP in the subnet range.
3. Host IPs:
Host IPs are the usable addresses within the subnet, excluding the network and broadcast addresses.

Q42. Match

	List – I
A.	AES
B.	RSA
C.	HMAC
D.	Digital

1. A–II, B–I
2. A–III, B–
3. A–II, B–I
4. A–IV, B–

Answer:
C
Sol: Let's ma primary uses:
A. AES: AES is used for symmetric e
Match: A →
B. RSA: RSA tion and key transport. It blic key for encryption a other use
Match: B →
C. HMAC: H nd message authenticati fication and authenticati
Match: C →
D. Digital Signature (e.g., ECDSA): Digital Signatures, such as ECDSA (Elliptic Curve Digital Signature Algorithm), provide non-repudiation and signer authenticity. They verify the identity of the sender and ensure that the message has not been altered.
Match: D → IV (Non-repudiation and signer authenticity)
Information Booster:
1. AES (Advanced Encryption Standard): AES is a symmetric encryption algorithm widely used in modern cryptography. It is efficient and secure, operating on fixed block sizes of 128 bits and supporting key sizes of 128, 192, and 256 bits.
2. RSA (Rivest-Shamir-Adleman): RSA is a widely-used public-key encryption algorithm that is essential in secure communications, often used in protocols such as HTTPS, for encrypting data and performing key exchanges.

3. HMAC (Hash-based Message Authentication Code): HMAC is used to ensure data integrity and authenticity, providing a mechanism to verify that the received message has not been tampered with and is indeed from the sender.
4. Digital Signatures (e.g., ECDSA): Digital signatures provide cryptographic proof of the authenticity of a message or document, ensuring that the signer cannot deny signing the message (non-repudiation), and validating the sender's identity.

Q.43 Arrange the steps in **mid-point circle algorithm** for rasterizing a circle:

1. Initialize decision parameter $p_0 = 1-r$.
2. Plot initial point (0, r) and symmetric points in all octants.
3. Increment x, update decision parameter.
4. If $p_k < 0, y_{k+1} = y_k$; else decrement y.
5. Repeat until x ≥ y.

A. 1 → 2 → 3 → 4 → 5
B. 2 → 1 → 3 → 4 → 5
C. 1 → 3 → 2 → 4 → 5
D. 2 → 3 → 1 → 4 → 5

Answer: A

Sol: The **Mid-Point Circle Algorithm** is a widely used algorithm for **rasterizing circles** on a grid-based display. It incrementally plots points on the circle's perimeter by calculating points in one octant and using symmetry to plot the points in all the other octants.

Step-by-Step Breakdown:

1. **Initialize decision parameter**: Start by initializing the **decision parameter** p_0 to 1 - r, where r is the circle's radius. This parameter helps in determining whether to increment or decrement the y coordinate as you progress through the algorithm.
2. **Plot the initial points**: Plot the initial point (0, r) on the circle and use the **symmetry** of the circle to plot points in all **eight octants** of the circle. This means you plot the points symmetrically at the positions derived from the original (0, r) point.

3. **Increment x and update decision parameter**: Increment the x coordinate to move along the perimeter of the circle. After each step, the decision parameter p_k is updated based on whether the previous point was inside or outside the circle.
4. **Conditionally adjust y**: If $p_k < 0$, the next point in the circle remains at the same y value as the previous point. If $p_k \geq 0$, it indicates the decision to move to a different y-coordinate (decrement the y value).
5. **Repeat until x ≥ y**: Continue this process, incrementing x and updating y until the x value becomes greater than or equal to the y value. This ensures that you have covered all necessary points on the circle.

Information Booster:

1. **Symmetry**: The algorithm takes advantage of the **symmetry** of the circle to compute only one octant and then uses that to plot the remaining points. This drastically reduces computation.
2. **Decision parameter**: The decision parameter p_k is key to determining the next pixel. If p_k is positive, the next pixel is in the outer part of the circle, and if it's negative, it's on the inner part of the circle.
3. **Efficiency**: The **Mid-Point Circle Algorithm** is efficient because it uses integer-only arithmetic (addition and subtraction), making it faster than methods that involve floating-point operations.

Additional Knowledge:

· **Symmetry of a circle**: The circle has eight octants (quarters of the plane), and plotting one octant and reflecting it across the axes is a key feature of efficient circle rasterization algorithms.
· **Rasterization**: This process of converting geometric shapes (like circles) into pixels on a grid is fundamental in computer graphics, especially in 2D rendering.

Q44. Which [illegible] ating a line between two [illegible]?

1. Digital D[illegible]tial Analyzer
2. Mid-Poi[illegible]
3. Bresenh[illegible]
4. Scan Lin[illegible]

Answer:
C
Sol:
The Bresenham's Line Algorithm is a famous algorithm used for drawing lines on digital displays. It is specifically designed to avoid floating-point operations, which makes it efficient for hardware implementations and pixel-based displays, where integer-only operations are typically preferred.
Key points of Bresenham's Line Algorithm:
The algorithm uses integer arithmetic (only additions and subtractions of integers) to calculate the points on the line, which avoids the computational overhead of floating-point operations.
The algorithm computes the next pixel to plot based on the decision variable, which is updated using simple integer operations.
This algorithm is fast and efficient because it avoids the use of complex mathematical calculations like multiplication and division.

Information Booster:
1. Efficiency: Algorithms like Bresenham's are preferred in graphics hardware because they avoid floating-point operations, which are slower than integer operations.
2. DDA vs. Bresenham's: DDA is simpler but slower due to the use of floating-point calculations, whereas Bresenham's avoids floating-point arithmetic and performs faster by using integer arithmetic.
3. Applications: Bresenham's Line Algorithm is widely used in computer graphics and displays for efficient rendering of lines, circles, and other shapes.
4. Related algorithms: The Mid-Point Circle Algorithm also avoids floating-point arithmetic and is optimized for drawing circles in a similar manner to how Bresenham's is optimized for lines.
Additional Knowledge:
Scanline Polygon Fill: This algorithm works by determining the parts of the screen to be filled based on the scanlines, and while it involves integer arithmetic, it doesn't directly minimize floating-point operations l
Rasterizatio
are used to convert vect

Q45. In line
subtraction without floa
1. Digital D
2. Bresenh
3. Midpoin
4. Flood-fil
Answer:
B
Sol:
Bresenham'
wn for its efficiency,
subtraction and bit-shif
e wh
to illu
e. It avoids floating-poi
. DDA uses floating-poi
Information
1. Bresenha
rit
n i
en
nta
y
os
pixel
sitions
2. It minimiz
putationa
ov
he
d
line
eri
ion.
3. DDA uses
ng
ri
metic
lower
older
ardware.
4. Midpoint
5. Flood-fill
6. Bresenha
7. It is widel
Additional Knowledge:
DDA approximates slopes using increments, less efficient.
Midpoint circle algorithm extends Bresenham's principles for curves.

Q46. Given below are two statements:
Statement I: Bezier curves are curves that interpolate all of their control points.
Statement II: A cubic bezier curve has four control points.
In the light of the above statements, choose the correct answer from the options given below:
1. Both Statement I and Statement II are correct.
2. Both Statement I and Statement II are incorrect.
3. Statement I is correct but Statement II is incorrect.
4. Statement I is incorrect but Statement II is correct.

Answer:
D
Sol:
Let's evaluate each statement:
Statement I: "Bezier curves are curves that interpolate all of their control points."
Incorrect. Bezier curves do not necessarily interpolate all of their control points. Instead, they are influenced by the control points, but they only pass through the first and last control points. The intermediate points are used to define the shape of the curve, but the curve does not necessarily pass through them (unless the degree of the curve is increased, or special cases are used).
Statement II: "A cubic Bezier curve has four control points."
Correct. A cubic Bezier curve has exactly four control points. The cubic Bezier curve is defined by four points: the two endpoints and two control points that influence the shape of the curve.
Information
1. Bezier Cu
o A Bezier c raphics and animation. T r curves are:
Linear Bezi
Quadratic B
Cubic Bezie
o The curve he co rder Bezier curves. It is
2. Cubic Bez
o A cubic Be ntrol nts, c 3, where P0 and P3 are t e the rol po e curve. The general form

$$B(t) = (1-t)^3 P0 + 3(1-t)^2 tP1 + 3(1-t)t^2 P2 + t^3 P3$$

The curve starts at P0 and ends at P3, and it is shaped by the control points P1 and P2.

Additional Knowledge:

- **Higher Degree Bezier Curves:** Higher-degree Bezier curves (such as quartic, quintic, etc.) can interpolate more control points. A Bezier curve of degree n will have n+1 control points. For example, a quartic Bezier curve will have 5 control points, and a quintic Bezier curve will have 6.

Q.47 Suppose a Bezier curve P(t) is defined by the following four control points in the xy − plane: $P_0 = (-2, 0)$; $P_1 = (-2, 4)$; $P_2 = (2, 4)$; and $P_3 = (2, 0)$. Then, which of the following statements are correct?

A. Bezier curve P(t) has degree 3.

B. $P\left(\frac{1}{2}\right) = (0, 3)$

C. Bezier curve P(t) may extend outside the convex hull of its control points.

Choose the correct answer from the options given below:

A. (A) and (B) only
B. (A) and (C) only
C. (B) and (C) only
D. (A), (B) and (C)

Answer: A

Sol:

1. **Understanding Bezier Curves:** A Bezier curve is defined by its control points. The degree of the curve is $n-1$, where n is the number of control points.
2. **Degree of the Bezier Curve:**
   - The curve is defined by 4 control points: P_0, P_1, P_2, P_3.
   - Degree $= 4-1 = 3$.
   - Statement A is **true**.
3. **Finding $P\left(\frac{1}{2}\right)$:**
   - A cubic Bezier curve is given by the equation:

$$P(t) = (1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3$$

Substituting $t = \frac{1}{2}$,

$$P\left(\frac{1}{2}\right) = \left(1-\frac{1}{2}\right)^3 P_0 + 3\cdot\frac{1}{2}\cdot\left(1-\frac{1}{2}\right)^2 P_1 + 3\cdot\left(\frac{1}{2}\right)^2\left(1-\frac{1}{2}\right)P_2 + \left(\frac{1}{2}\right)^3 P_3$$

Simplify each term:

- $\left(1-\frac{1}{2}\right)^3 P_0 = \frac{1}{8}P_0 = \frac{1}{8}(-2,0) = \left(-\frac{1}{4},0\right)$,
- $3\cdot\frac{1}{2}\cdot\left(1-\frac{1}{2}\right)^2 P_1 = \frac{3}{8}P_1 = \frac{3}{8}(-2,4) = \left(-\frac{3}{4},\frac{12}{8}\right) = \left(-\frac{3}{4},1.5\right)$,
- $3\cdot\left(\frac{1}{2}\right)^2\cdot\left(1-\frac{1}{2}\right)P_2 = \frac{3}{8}P_2 = \frac{3}{8}(2,4) = \left(\frac{3}{4},1.5\right)$,
- $\left(\frac{1}{2}\right)^3 P_3 = \frac{1}{8}P_3 = \frac{1}{8}(2,0) = \left(\frac{1}{4},0\right)$.

Summing these terms:

$$P\left(\frac{1}{2}\right) = \left(-\frac{1}{4},0\right) + \left(-\frac{3}{4},1.5\right) + \left(\frac{3}{4},1.5\right) + \left(\frac{1}{4},0\right) = (0,3)$$

Statement B is **true**.

4. **Convex Hull Property:** Bezier curves always lie within the convex hull of their control points. Statement C is **false**.

Information Booster:

1. **Degree of Bezier Curves:** The degree is always $n-1$, where n is the number of control points.
2. **Applications:** Bezier curves are widely used in computer graphics, animation, and path modeling.

Additional Knowledge:

- **Control Points:** Control points influence the shape of the curve, but the curve doesn't pass through all of them (except for the endpoints).
- **De Casteljau's Algorithm:** A recursive algorithm for evaluating Bezier curves at any point t.

Q48. Match List-I with List-II

	List – I (Graphics Concepts)		List – II (Descriptions/Techniques)
A.	Ray Tracing	i.	Technique for hidden surface removal
B.	Gouraud Shading	ii.	Smooth shading using vertex intensities
C.	Z-Buffer Algorithm	iii.	Simulates realistic lighting and shadows
D.	Phong Reflection Model	iv.	Calculates intensity per pixel for realism

Match the columns.

1. (A)-(iii), (B)-(ii), (C)-(i), (D)-(iv)
2. (A)-(iv), (B)-(iii), (C)-(ii), (D)-(i)
3. (A)-(i), (B)-(iv), (C)-(iii), (D)-(ii)
4. (A)-(ii), (B)-(i), (C)-(iv), (D)-(iii)

Answer:

A

Sol:

(A) Ray Tracing → (iii):

Ray tracing is a technique that simulates realistic lighting and shadows by tracing the path of rays of light as they interact with objects in the scene.

It handles reflections, refractions, and shadows accurately.
Example: Used in movie animations and games for high-quality rendering.
(B) Gouraud Shading → (ii):
Gouraud shading is a smooth shading technique that interpolates vertex intensities across the surface of a polygon.
It provides a smooth appearance but may miss specular highlights.
(C) Z-Buffer Algorithm → (i):
The Z-buffer algorithm is a technique for hidden surface removal.
It maintains a depth buffer to track the closest object for each pixel.
Example: Used in rendering pipelines to resolve depth conflicts.
(D) Phong Reflection Model → (iv):
The Phong reflection model calculates intensity per pixel, considering ambient, diffuse, and specular components for realistic shading.
Unlike Gour
Information
1. Ray Traci te shadows, reflections,
2. Shading T
Flat Shading
Gouraud Sh nterpo
Phong Shad l intens
3. Hidden Su
Z-Buffer Alg
Painter's Alg st to st.
Scanline Alg pixels at a time.
4. Phong vs.
Gouraud: Fa hlight
Phong: Slow
Additional K
Applications racin
Real-time re in od n Us or me (RTX).
High-end VF ovies.
Importance uff sential for solving visibilit problems in 3D graphics pro .
Shading in for realistic rendering in

Q49. Match

List – I (IP Address)	List – II (Class)		
A.	10.20.30.40	I.	Class E
B.	210.20.30.3	II.	Class B
C.	180.30.100.10	III.	Class A
D.	252.5.15.11	IV.	Class C

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

Answer:
D

Sol:
1. Class A: 0.0.0.0 to 127.255.255.255 (e.g., 10.20.30.40).
2. Class B: 128.0.0.0 to 191.255.255.255 (e.g., 180.30.100.10).
3. Class C: 192.0.0.0 to 223.255.255.255 (e.g., 210.20.30.3).
4. Class E: 240.0.0.0 to 255.255.255.255 (e.g., 252.5.15.11).
Information Booster:
IP Address Classes: IP addresses are divided into classes (A, B, C, D and E) based on their range, used primarily to organize networks.

Q50. COCOMO is _____, Function Point is _____, Delphi is _____, and Putnam Model is _____.
1. Algorithmic, Metric-based, Expert-based, Statistical distribution
2. Expert-based, Algorithmic, Metric-based, Statistical distribution
3. Metric-b
4. Algorithi

Answer:
A
Sol:
COCOMO (C ns based on project size
Function Po quantifying functionality
Delphi Tech sing i erati
Putnam Mo istrib on (R t and time.
Thus, the co gori Func hi → Expert-based Putna tion
Information
1. COCOMO sing formu
2. Function
3. Delphi us ip ert
4. Putnam M lies t ayleigh istribu on for
5. Each repr di ere ap oa t oft are est ation.
Additional K dge:
(b) Expert- Metr based Statist cal distribution. Wro , be COCOMO is algorithmic,
(c) Metric-b nction Point and Putnam
(d) Algorith nt is metric-based, not expert-based

Q51. Consider the following statements regarding the COCOMO (Constructive Cost Model):
I. COCOMO is used to estimate the cost, effort, and schedule of software projects.
II. The model is divided into three levels: Basic, Intermediate, and Advanced.
III. COCOMO does not consider factors like team experience or hardware constraints.
IV. The Intermediate COCOMO level uses cost drivers to refine estimates.
V. The Advanced COCOMO incorporates detailed factors like reuse and documentation.
Which of the above statements are TRUE?
1. I, II, IV and V
2. I, III and IV
3. II, III and IV

4. I, II and III

Answer:

A

Sol:

Statements I, II, IV and V are correct as they reflect the key aspects of the COCOMO model: its usage, levels, and refinements at different stages. However, statement III is false because the Intermediate and Advanced levels of the model do consider factors like team experience, hardware and other constraints, which significantly impact the effort and cost estimation.

Information Booster:

1. COCOMO's Purpose: The COCOMO model is widely used for estimating effort, cost, and time in software development projects.

2. Three Levels:

Basic: Provi

Intermediat

Advanced: A

3. Cost Driv mental and technical fac

Additional K

Basic COCOM d is lex systems.

Intermediat ncorp ce, product complexity a

Advanced C ailed cost x projects. It includes cor ents stem eeds.

[redacted]

Q.52 According to **Basic COCOMO Model**, the development effort for **semi-detached software** is given by:

A. $E = 3.0 \times (KLOC)^{1.12} PM$

B. $E = 3.0 \times (KLOC)^{1.05} PM$

C. $E = 2.4 \times (KLOC)^{1.05} PM$

D. $E = 2.8 \times (KLOC)^{1.12} PM$

Answer: B

Sol: In the COCOMO (Constructive Cost Model), there are different models based on the type of software project being developed. In the **Basic COCOMO Model**, the development effort equation depends on the project category: **organic**, **semi-detached**, or **embedded**. For **semi-detached software projects**, which have a mix of experienced and inexperienced developers and moderate complexity, the standard effort estimation formula is:

$E = 3.0 \times (\text{KLOC})^{1.12}\ \text{PM}$

This formula reflects higher complexity than organic projects but less rigidity than embedded ones. Hence, option **(a)** correctly represents the Basic COCOMO effort equation for semi-detached projects.

Information Booster:

1. **Basic COCOMO Model:** An empirical software cost estimation model proposed by Barry Boehm.
2. **Semi-detached projects:** Medium-sized projects with moderate constraints and mixed team experience.
3. **Effort unit:** Measured in **Person-Months (PM)**.

4. **Exponent significance (1.12):** Indicates super-linear growth of effort with size due to increasing complexity.
5. **Coefficient (3.0):** Calibrated constant specific to semi-detached projects.
6. **Usage:** Provides early-stage effort estimation when only KLOC is known.

Additional Knowledge:

- **Reference formulas (Basic COCOMO):**
  - **Organic:** $E = 2.4 \times (\text{KLOC})^{1.05}$
  - **Semi-detached:** $E = 3.0 \times (\text{KLOC})^{1.12}$
  - **Embedded:** $E = 3.6 \times (\text{KLOC})^{1.20}$

Q53. Which
1. COCOMO
2. Function
3. Wideban
4. Use Case
1. Only 1 a
2. Only 2 a
3. Only 1, 2
4. Only 2, 3
Answer:
C
Sol:
Model-base … math … or emp … measurable software at … d schedule. Among the g … ons …) I … tic … nalysis … th … tion), … Case Points are all mod … tech … es, as … ey re … on … i … formula … a … ram … t … erived from historical or … cal … ata … n c … tra … t, … de … Del … is an e … pert-ju … g … nt … e … not a model-based appro
Information … ter
1. COCOMO
Developed b
Formula: Ef
Provides eff
2. Function Point Analysis (FPA):
Quantifies system functionality by counting inputs, outputs, interfaces, and data files.
With calibration, adjusts values based on environmental and technical complexity factors.
3. Use Case Points (UCP):
Estimates effort using the number and complexity of use cases and actors.
Includes technical and environmental weightings for accuracy.
4. Model-Based Estimation:
Relies on empirical data and statistical modeling.
Produces repeatable, objective estimates compared to expert-driven methods.
5. Applications: Used during project planning and feasibility analysis to predict effort, cost, and schedule.
6. Accuracy Improvement: Achieved through organization-specific calibration using past project data.

7. Tools: Estimation models are often implemented in software like SEER-SEM, SLIM, or Costar.
Additional Knowledge:
(1) COCOMO II: Model-based; quantitative model with empirical equations.
(2) Function Point Analysis: Model-based when calibrated; relies on weighted functional metrics.
(3) Wideband Delphi: Expert-based; uses consensus estimation among domain experts, not mathematical modeling.
(4) Use Case Points: Model-based; mathematically derives effort from quantified use case metrics.
Model-based approaches depend on quantitative formulas calibrated from real data, making them more objective and scalable than judgment-based methods like Delphi.

Q.54 Given the following **COCOMO model estimation parameters**:

- Estimated number of lines of code (LOC): 50,000
- Cost driver: Organic (with a multiplier of 2.4)
- The **Basic COCOMO** formula: $\text{Effort} = a \times (\text{KLOC})^b$

What is the **estimated effort** (in person-months) using the Basic COCOMO model, where a = 2.4 and b = 1.05?

A. 124.83 person-months
B. 98.74 person-months
C. 148.56 person-months
D. 85.91 person-months

Answer: A

Sol: Given:

- Estimated number of lines of code (LOC) = 50,000
- Cost driver: Organic (with a multiplier of 2.4)
- Basic COCOMO formula:

$\text{Effort} = a \times (\text{KLOC})^b$

where:

- a = 2.4 (for Organic type project)
- b = 1.05

Step 1: Convert LOC to KLOC

COCOMO uses KLOC (Kilo Lines of Code), which is the number of thousands of lines of code.

$$\text{KLOC} = \frac{\text{LOC}}{1000}$$

Substitute the value of LOC:

$$\text{KLOC} = \frac{50,000}{1000} = 50$$

Step 2: Apply the Basic COCOMO formula

Now, use the Basic COCOMO formula to calculate the effort:

$\text{Effort} = a \times (\text{KLOC})^b$

Substitute the given values:

$\text{Effort} = 2.4 \times (50)^{1.05}$

Step 3: Calculate $(50)^{1.05}$

To calculate the exponent:

$(50)^{1.05} \approx 52.015$

Step 4: Calculate the final effort

Now, multiply the result by a = 2.4:

$\textit{textEffort} = 2.4 \times 52.015 \approx 124.83$ person-months

Q55. Given a project that uses the COCOMO model with an estimated effort of 2000 person-months and a productivity rate of 5 person-month per KLOC, what is the estimated size of the project in KLOC?

1. 200
2. 400
3. 100
4. 50

Answer:

B

Sol:

COCOMO Formula:

Size (KLOC) Effort Productivity

Calculation: Size = 2000 / 5 = 400KLOC

Information

COCOMO M[redacted]gineering.

Q.56 Estimation of software development effort for organic software in basic COCOMO is:

A. $E = 2.4 \times (\text{KLOC})^{1.05}$ PM
B. $E = 3.4 \times (\text{KLOC})^{1.06}$ PM
C. $E = 2.0 \times (\text{KLOC})^{1.05}$ PM
D. $E = 2.4 \times (\text{KLOC})^{1.07}$ PM

Answer: A

Sol: The **COCOMO (Constructive Cost Model)** is a software cost estimation model that helps in estimating the effort required for software development. It uses the **KLOC (Kilo Lines of Code)** as the input parameter to estimate the **effort** in **person-months (PM)**.

In **basic COCOMO**, there are three types of software projects:

1. **Organic**: Small, simple software projects with a small team of developers.
2. **Semi-Detached**: Intermediate-sized software projects.
3. **Embedded**: Large, complex software projects with strict requirements.

The formula for estimating software development effort in **organic software** using the **basic COCOMO model** is:

$$E = 2.4 \times (\text{KLOC})^{1.05} \text{ PM}$$

Where:

- E is the effort in person-months (PM),
- KLOC is the number of thousands of lines of code (Kilo Lines of Code),
- The constants 2.4 and 1.05 are derived from empirical data for organic software projects.

Given:

- The formula for organic software in basic COCOMO is:

$$E = 2.4 \times (\text{KLOC})^{1.05} \text{ PM}$$

Information Booster:

1. COCOMO (Constructive Cost Model): COCOMO is a software estimation model developed by Barry Boehm in the 1980s. It is used to estimate the effort, time and cost required for software development based on the size of the software (measured in KLOC) and various cost drivers.
2. COCOMO Model Versions:

Basic COCOMO: Provides a simple estimation based only on KLOC (lines of code).

Intermediate COCOMO: Uses additional factors (cost drivers) such as product complexity, hardware constraints and developer experience.
Detailed COCOMO: Adds even more detailed factors and provides a more precise estimate.
3. Organic Software: Organic software refers to small, well-understood software projects typically developed by a small team. The basic COCOMO model assumes that the software is small, relatively simple and does not require much coordination between a large number of developers.
4. KLOC (Kilo Lines of Code): The unit "KLOC" refers to thousands of lines of code and is a common metric used to measure the size of a software project. For example, a 50 KLOC project means that the software has 50,000 lines of code.
Additional Knowledge:
Person-Month (PM): A person-month is a unit of work, representing the amount of effort one person would expend in one month. In software development, it is used to measure the total effort required to complete th
Cost Drivers
cost drivers, which inclu
Developer e
Hardware c
Required re
Complexity

Q57. Arrang
der of co
est coupling:
A. Common
B. Stamp Co
C. Control C
D. External
E. Content C
Choose the
ions gi
1. E, A, C, B
2. D, B, A, E
3. B, C, D, A
4. C, A, B, D
Answer:
C
Sol:
In software
re modules.
The lower th
The correct
akest to the strongest within the given set. Stamp coupling is the least intrusive here—modules share a composite record/struct and typically use only parts of it. Control coupling is stronger since one module steers another's logic via flags or control parameters. External coupling binds modules to outside formats/devices/interfaces, increasing rigidity beyond mere control flow. Common coupling shares global data, creating broad, hard-to-track dependencies. Content coupling is the strongest (worst), where a module reaches into another's internals or its code structure.
Information Booster
1. Definition Ladder (weak → strong in this list): Stamp (B) → Control (C) → External (D) → Common (A) → Content (E).
2. Stamp Coupling (B):
   - Type: Share a whole record/struct; callee uses only some fields.
   - Example: Passing Customer when only Customer.id is used.

- Applications/Uses: Common in APIs where DTOs are passed; acceptable when records are stable.
- Advantages: Lower logical dependency than control/common/content; clearer interfaces than globals.
- Disadvantages: Over-passing data can hide what's truly needed; potential for unintended future use of extra fields.

3. Control Coupling (C):
- Type: Caller sends flags/modes that alter callee's internal flow.
- Example: processOrder(order, isExpress=true).
- Applications/Uses: Mode switches, feature toggles.
- Advantages: Single entry point for multiple behaviors.
- Disadvantages: Logical entanglement; callee behavior depends on external control knowledge.

4. External Coupling (D):
- Type: De variables.
- Example r spec.
- Applicat
- Advanta
- Disadva

5. Commo
- Type: M dat
- Example tten b
- Applicat rototypi
- Advanta
- Disadva testa ty, tig m.

6. Content
- Type: Or nother's internal
- Example er class's private fie nory pokes.
- Applicat seen i hacks
- Advanta
- Disadva eak lat ax ly, ma ce ntma

7. Practica m & ects): P fer int faces a o y ed d ta (a pling, not in options). If y st ss tru k ep sta e and d cumen (stam vo d flags when separate fu or strateg pa erns ca exp nt . Treat s a t r er reach into internals.

Additional K

Lower cou ability, and maintainabi

High coupli s harder to manage.

Q58. Which of the following statements about cohesion and coupling is correct?
1. High cohesion within a module improves maintainability.
2. Low coupling between modules reduces dependency.
3. High coupling improves modularity.
4. Low cohesion is preferred for reusable components.

1. 1 and 2 are correct
2. 1 and 3 are correct
3. 2 and 4 are correct
4. All statements are correct.

Answer:

A
Sol:
Let's break down the statements one by one:
Statement 1: "High cohesion within a module improves maintainability."
Correct. High cohesion means that the elements within a module are closely related in functionality. A module with high cohesion is easier to understand, maintain, and modify, since all its components work towards a single, well-defined purpose. This improves maintainability and reduces the likelihood of introducing errors when making changes.
Statement 2: "Low coupling between modules reduces dependency."
Correct. Low coupling means that modules do not depend heavily on each other. When modules are loosely coupled, they can function independently, which reduces dependencies and makes it easier to modify or replace one module without affecting others. This also enhances modularity and simplifies testing and maintenance.
Statement 3
Incorrect. H his reduces modularity, gh coupling leads to a m
Statement 4
Incorrect. L es it harder to understa typic form a well-defined an increases because it encapsulate s depende
Information
1. Cohesion resp ilities are. A highly cohesive mo urpose, making it use.
2. Coupling idence between mo d because it makes the s e, and maint changes and allows for b

Q59. Match th Li .

List – I (So design pri s)	**ist II De ition)**		
A.			another
B.			ct modules.
C.			belong
D.	Modularity	IV.	Simplifying complex reality by modeling classes appropriate to the problem.

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

Answer:
C
Sol:
A. Cohesion (III: Degree to which elements of a module belong together)

Cohesion refers to how closely related and focused the responsibilities of a single module are. It's a measure of the strength of relationship between elements within a module, which matches Definition III.

B. Coupling (I: Degree to which one module relies on another module)

Coupling indicates the level of dependency between modules. High coupling means modules are highly dependent on each other, while low coupling means they are more independent. This aligns with Definition I.

C. Abstraction (IV: Simplifying complex reality by modeling classes appropriate to the problem)

Abstraction is a design principle that focuses on hiding complex details and showing only the necessary aspects of an object, which allows a simpler view of the reality being modeled. This matches Definition IV.

D. Modularity (II: Dividing a software system into distinct modules)

Modularity is the practice of dividing a software system into separate components or modules that can be develope

Q60. Alpha

1. White –
2. Black – B
3. Acceptan
4. System T

Answer:

C

Sol:

Alpha and B ing t ques vare against user needs a yment.

Information

1. Alpha Tes team lled e s and gather feedback.
2. Beta Test u r vironmen to cover y issues and ensure satis
3. Purpose: th oftwar eets b ine s and use requir ments e e r a
4. Types of A nce Testi g: I ludes r acc pan e testing (, op ti a ance testing, and contrac sti
5. Outcome:

Additional K

White-box t

Black-box te

System testing evaluates the entire integrated system.

Q61. Which statement correctly differentiates between alpha and beta testing?

1. Alpha testing is done by end users, whereas beta testing is done by developers.
2. Alpha testing is performed after release, while beta testing is done during coding.
3. Alpha testing occurs in a controlled environment; beta testing takes place at user sites.
4. Beta testing includes unit testing whereas alpha testing does not.

Answer:

C

Sol:

Alpha testing is typically done by the development team or internal testers in a controlled environment before the product is released to external users. It focuses on finding bugs and issues in the software

that can be addressed before release. On the other hand, beta testing is conducted by end users at user sites, allowing them to test the product in a real-world environment. The goal of beta testing is to gather feedback on the software's functionality, usability and performance in actual user conditions.

Information Booster

1. Alpha Testing:

Performed in-house by developers or quality assurance (QA) testers.

Occurs before the product release and is part of the internal testing phase.

It's a controlled environment where the testing team checks for bugs, usability issues, and functionality problems.

Involves internal testers, which could include developers, QA teams, or sometimes selected employees of the organization.

2. Beta Testing:

Performed b

Occurs after

minor bugs.

The objectiv

w it behaves in practical

Helps gathe

at were not caught duri

3. Other Opt

(a) is incorr

house

users. Beta testing is wh

(b) is incorr

befo re re

s done after alpha testin

(d) is incorr

t inv

nit t

a testing or earlier stag

focusing on testi

units of the software.

Additional K

Alpha testin

like system integration,

Beta testing

al us

to prov

e feedb

ck on

a

perfo

nan

d i

s

at might not have been a

ed

th

dev

opm

ea

Both alpha

ta testing re

porta

phase

n th

softwa

ease

ec

ing that the product is r

and

fr

ndly b

ore its

nal laun

Q62. Match

	List – I		
A.	Alpha T		
B.	Beta Testing	II.	Performed by actual customers.
C.	Regression Testing	III.	Ensures new code doesn't break old functionality.
D.	Unit Testing	IV.	Testing by end users at development site.

1. A – I, B – II, C – III, D – IV
2. A – II, B – I, C – IV, D – III
3. A – III, B – II, C – I, D – IV
4. A – IV, B – II, C – III, D – I

Answer:

D

Sol:

Each type of software testing has a specific focus and phase. Alpha testing is done by real users but inside the developer's premises. Beta testing takes place in the customer's environment. Regression

testing ensures that recent changes haven't introduced new bugs in previously working modules. Unit testing is a white-box technique used to verify the correctness of individual program units.
Information Booster:
1. Alpha Testing: Internal acceptance testing by user representatives.
2. Beta Testing: External acceptance testing in user's environment.
3. Regression Testing: Prevents side effects due to code changes.
4. Unit Testing: Tests each program component in isolation.

Q63. In Basis Path Testing, if a program's control flow graph has 8 nodes, 10 edges and 1 connected component, what is the Cyclomatic Complexity?
1. 3
2. 4
3. 5
4. 6

Answer:
B
Sol:
The Cyclom a program's
control flow r of e program's
source code g form
V(G) = E - N
Where:
E is the num graph
N is the num gra
P is the num in the graph.
Given:
Nodes (N) =
Edges (E) =
Connected c s (
Now, applyi mula

$$V(G) = 10 - 8 + 2 \times 1 = 10 - 8 + 2 = 4$$

Thus, the **Cyclomatic Complexity** of the program is **4**.

Information Booster:

1. **Cyclomatic Complexity**: It helps in determining the complexity of a program by counting the number of linearly independent paths. Higher complexity indicates that the program has more decision points, making it harder to test and maintain.
2. **Formula for Cyclomatic Complexity**: V(G) = E - N + 2P is derived from graph theory and reflects the number of independent paths in the control flow graph.
3. **Control Flow Graph (CFG)**: The **nodes** represent program statements or decisions, and the **edges** represent the control flow between them. Cyclomatic complexity helps in determining how many test cases are needed to achieve full path coverage.

Additional Knowledge:

Cyclomatic complexity is widely used in **software testing** to ensure adequate test coverage. It is especially useful for determining the minimum number of paths that must be tested to achieve full coverage.

Q64. When estimating software projects using Lines of Code (LOC) and Function Points (FP), which of the following statements is true?
A. LOC measures the size of a software project based on the number of lines of code written.
B. Function Points provide a more accurate estimation of effort compared to LOC.

C. LOC estimation is independent of the complexity and functionality of the software.
D. Function Points can only be calculated after the software has been fully developed.
Choose the correct answer from the options given below:
1. A, B and C only
2. B only
3. A, C and D only
4. A, B and D only
Answer:
B
Sol:
A. LOC (Lines of Code) measures the size of a project based on the number of lines of code written. While it provides some insight into the size of the code, it is not the most effective for estimating effort because it does not a
B. Function eveloping a software sys account the complexity like inputs, outputs, inq
C. LOC estim ble software system may nes, making LOC an inac
D. Function software is fully develop LOC, which requires the
Information
1. Lines of C measure softwar ch as:
It does not plexity of the code.
It can incen d be seen as an indicator
It depends
2. Function P):
o Function P considering inputs, outp data stor
o Function estimation before codi
o They prov ware is from a functional
3. Comparis
o LOC tends to be biased towards certain languages and does not always correlate well with complexity or functional effort.
o FPs provide a more consistent measurement of a software project's effort across different languages and platforms, making them better suited for early project estimation and cross-project comparisons.
4. Effort Estimation in Software Development: Using function points gives a more predictive understanding of the effort involved in developing software. It can also be used to track progress and manage project scope over time.
Additional Knowledge
• Function Points for Agile Methodologies: In Agile projects, function points can help measure story points and track how much effort is involved in implementing a feature from a functional perspective, which helps in sprint planning.

• Software Metrics: Software metrics, such as LOC and FP, are used to assess the efficiency, complexity and progress of development. However, it's crucial to combine multiple metrics, such as defect density or customer satisfaction, to get a comprehensive picture of software quality.

Q65. The V-Model of software development is not ideal for projects where:
1. Requirements change frequently.
2. Each phase corresponds to a testing activity.
3. Proper verification and validation are essential.
4. The project follows a sequential approach.

Answer:
A
Sol:
The V-Model (Verification and Validation Model) is a sequential software development model derived from the Waterfall Model. It is ideal for projects where the development phase is directly linked to the testing phase, and requirements are fixed at the beginning of the project. It is not suitable for projects where requirements are unstable or may change frequently.
Information
1. The V-Mo req anges.
2. It provide alignm
3. It is not fl e or rapi
Additional K
Each phase ity: T is a – for every developmen ing p
Proper verif ential. The V-Mode sure both — hence, this i a limitation.
The project ch: Thi desc ; sequential flow is its fu

Q66. Match ving:

	List – I re roc s		List – II (Charac ris
	Models		
A.	Waterf		A mod l where each phase must be comple ore the
B.	Spiral		during
C.	V-Model	III.	e development phase.
D.	Agile Model	IV.	Encourages flexibility and iterative progress with customer feedback.

1. A → I, B → II, C → III, D → IV
2. A → II, B → I, C → III, D → IV
3. A → III, B → II, C → I, D → IV
4. A → IV, B → III, C → I, D → II

Answer:
A
Sol:
Let's match the software process models with their characteristics:

1. Waterfall Model (A): I - The Waterfall model is a linear, sequential development process. Each phase of development must be completed before moving on to the next phase, and there is no iteration back to previous phases. This characteristic match "A model where each phase must be completed before the next starts."
2. Spiral Model (B): II - The Spiral model focuses on iterative development with a strong emphasis on risk analysis. The development progresses through repeated cycles (spirals), each including risk analysis, planning, and refinement of the product. This matches "Focuses on iterative risk analysis and refinements during development."
3. V-Model (C): III - The V-Model emphasizes validation and verification. Testing is planned and executed in parallel with each development phase, ensuring that each phase is tested as it is completed. This matches "Emphasizes testing at each stage in parallel with the development phase."
4. Agile Model (D): IV - The Agile model emphasizes flexibility, iterative progress, and collaboration with customers t ted into the developmen Encourages flexibility ar
Information
1. Waterfall s with well-defined rec ects where requiremen
2. Spiral Mo y usef projects. Its iterative nat ment, b d resources for risk anal
3. V-Model: the terfa early in the developmen ed in y-cri nd medical software, wh are crucial.
4. Agile Mo ke Scrum, Kanban) incremental updates thr ts). It f tinuo adjustments to changes i

Q67. Identif rect s ments f om the ollown lin spi al
A. Spiral mo in en ta ro ess od
B. Spiral mo risk-driv n p ces m le.
C. Spiral mo an s b lancing pproac
D. Spiral mo
Choose the
1. A and B
2. B and C
3. A and D only
4. B and D only

Answer:
D
Sol:
The Spiral Model is a software development process model that combines elements of both design and prototyping. Let's examine each of the statements given:
A. Spiral model is an incremental process model: Incorrect. While the spiral model does involve iterations, it is not strictly incremental. The spiral model emphasizes risk analysis and management at each iteration. It's a risk-driven model, but it doesn't strictly follow the incremental development approach like some other models (e.g., Agile). The process involves repeating the phases of planning, design, and testing, but these do not follow a simple incremental pattern.

B. Spiral model is a risk-driven process model: Correct. The Spiral Model is risk-driven. This means that it places a significant emphasis on risk management. Each iteration or "spiral" of the model involves assessing and managing risks, which is a key feature of the model.
C. Spiral model is an acyclic balancing approach: Incorrect. The term "acyclic balancing" does not apply to the spiral model. The spiral model is cyclic, not acyclic, because it involves repeated cycles (spirals) of development, with each cycle building upon the last. Therefore, it doesn't fit the description of an acyclic approach.
D. Spiral model involves a set of anchor point milestones: Correct. The Spiral Model uses anchor points, which are milestones in the development process where important deliverables are reviewed. These milestones help ensure that each iteration addresses the most critical aspects of the project, especially the risk assessment, planning, and design phases.
Information Booster:
1. Risk-driv ng potential risks in the
2. Anchor p hat specific deliverables
3. Cyclic de rom strictly incremental ent.

Q68. Spiral
1. Outward
2. Inner an
3. Outward
4. Inner clo

Answer:
A
Sol:
The Spiral M he waterfall model and i presenting a phase in the The spiral st ard denotes an incremer
Information
1. Expansio cement and growth.
2. Rotation g stages like objective se
3. Phased Iteration: Each loop through the spiral represents a new phase or iteration of the software with evolving functionality.
4. Risk Handling: One of the main strengths of the spiral model is its emphasis on risk analysis at every iteration.

Q69. Match the following:

	List – I (Process Models)		**List – II (Characteristics)**
A.	Spiral Model	I.	Rigid structure with sequential phases.
B.	V-Model	II.	Uses risk analysis and prototyping.
C.	Waterfall Model	III.	Incorporates validation and verification.
D.	Incremental Model	IV.	Delivers partial systems in builds.

1. A–II, B–III, C–I, D–IV
2. A–III, B–II, C–IV, D–I
3. A–IV, B–II, C–I, D–III
4. A–II, B–I, C–III, D–IV

Answer:
A
Sol:
Spiral Model uses risk analysis and prototyping to iteratively develop the system.
V-Model is an extension of the Waterfall model that emphasizes validation and verification at each development phase.
Waterfall Model is a rigid, sequential structure where each phase follows the other in order.
Incremental Model develops and delivers partial systems in builds, progressively adding functionality.
Hence, the c
Information
1. Spiral Mo
2. V-Model l
3. Waterfall
4. Incremen

Q70. Which ngine tal models?
1. WebE m for all la
2. WebE ex men infor engineering into the lifec
3. WebE di onte tility
4. WebE re by default.

Answer:
B
Sol:
Web Engine eb fr ene ental ode ts no proach that combines te devel ent wit conte mana in orm tic hit t nd usability engineering ho th roj ct ife le. nlike tandard ncremen a mo s ch primarily emphasize s e function lity eli ery, WebE ogn es that pp ion r ent-rich and user-centric f it int rates ultidis ciplinary processes involving content, and usability to
Information
1. Web Eng ches to web application
2. It includes elements from software engineering, information architecture, content management, and human-computer interaction (HCI).
3. WebE covers the entire lifecycle, from requirements and design to deployment, testing, and maintenance.
4. Unlike traditional models, it addresses usability, accessibility, and navigational design as core engineering activities.
5. WebE supports incremental and agile delivery, accommodating frequent updates and evolving content.
6. It ensures the balance between functionality and user experience, which is critical in web-based systems.
7. WebE also emphasizes security, scalability, and content version control, often missing in generic models.

Additional Knowledge:
(a) WebE mandates model-driven architecture for all layers: Incorrect — WebE can use model-driven techniques, but it doesn't mandate them universally.
(c) WebE disallows agile iterations: False — WebE encourages iterative and agile methods to manage frequent content and design updates.
(d) WebE requires serverless deployment by default: Incorrect — Deployment architecture (server-based or serverless) is a technical choice, not a WebE requirement.
What sets Web Engineering apart is its integration of content, structure, and usability into the engineering process — making it far more multidisciplinary and user-oriented than generic incremental approaches.

Q.71 The time complexity of an algorithm T(n), where n is the input size, is given by:

$$T(n) = T(n-1) + \frac{1}{n}, \quad \text{if } n > 1$$
$$T(1) = 1, \quad \text{otherwise}$$

The order of this algorithm is:

A. $\log n$
B. n
C. n^2
D. n^n

Answer: A

Sol: The given recurrence relation is:

$$T(n) = T(n-1) + \frac{1}{n} \quad \text{for } n > 1, \quad T(1) = 1$$

To solve this recurrence, we can expand it step by step:

$$T(n) = T(n-1) + \frac{1}{n}$$
$$T(n-1) = T(n-2) + \frac{1}{n-1}$$
$$T(n-2) = T(n-3) + \frac{1}{n-2}$$
$$\vdots$$

Continuing this process, we get:

$$T(n) = T(1) + \left(\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right)$$

Since T(1) = 1, we can rewrite the recurrence as:

$$T(n) = 1 + \left(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$$

The sum $\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ is the harmonic sum, which is approximately $\log n$ for large n.
Thus, the overall time complexity is $O(\log n)$.

Information Booster

1. The **harmonic series** $\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)$ grows logarithmically, meaning it is approximately equal to $\log n$ as n increases.

2. The recurrence is a classic example of problems where the time complexity can be determined by summing the terms of a harmonic series.

3. In general, if a recurrence relation involves subtracting 1 from the input size and adding a decreasing function (like $\frac{1}{n}$), the result often leads to logarithmic growth.

4. $\log n$ complexity is much better than linear O(n), quadratic $O(n^2)$, or exponential $O(n^n)$ time complexities for large inputs.

5. For algorithms with this recurrence, like certain divide-and-conquer algorithms, the $\log n$ complexity indicates efficient scaling with larger problem sizes.

6. The **harmonic sum** grows slowly, meaning the algorithm will run efficiently even for large input sizes.

7. In many real-world applications, algorithms with $O(\log n)$ complexity are preferred for tasks like searching, sorting and data retrieval.

Additional Knowledge

· **Option (b)** is incorrect because the recurrence relation does not grow linearly with n, but instead grows logarithmically.

· **Option (c)** is incorrect because n^2 would imply quadratic growth, which is much faster than logarithmic growth.

· **Option (d)** is incorrect because n^n growth is exponential, which is far too large for this recurrence relation.

Q72. Consid
(1) Depth-fi
(2) Pre-orde ee.
(3) Huffman ights.
(4) Topolog eir children.
Which of the statements are true?
1. (1) and (
2. (3) and (
3. (1), (2) a
4. (1), (2),

Answer:
C
Sol:
Step 1: Analyze Each Statement
Statement (1): Depth-first search is used to traverse a rooted tree – True: Depth-first search (DFS) is commonly used to traverse a tree. It explores as far as possible along each branch before backtracking, making it suitable for tree traversal.
Statement (2): Pre-order, Post-order, and In-order are used to list the vertices of an ordered rooted tree – True: Pre-order, Post-order, and In-order traversals are standard methods to list vertices of a rooted tree in specific orders, especially when the tree is ordered.
Statement (3): Huffman's algorithm is used to find an optimal binary tree with given weights – True: Huffman's algorithm constructs an optimal binary prefix tree (Huffman Tree) based on a set of weights (frequencies), minimizing the total weighted path length.

Statement (4): Topological sorting provides a labeling such that the parents have larger labels than their children – False: Topological sorting is applied to directed acyclic graphs (DAGs), not necessarily trees, and it provides an ordering of vertices such that for every directed edge u → v, u appears before v. It does not guarantee that parents have larger labels than their children in a tree structure.
Step 2: Conclusion
Statements (1), (2) and (3) are true.
Statement (4) is false.
Thus, the correct answer is (c): (1), (2) and (3).
Information Booster
1. Tree Traversals:
Pre-order: Visit root → left → right.
In-order: Visit left → root → right.
Post-order: Visit left → right → root
2. Huffman's [illegible] by creating a tree based [illegible]
3. Topologic[illegible]xist. Applied to DAGs but [illegible]

Q73. Match [illegible]

	List –		
A.	Circul[illegible]	I.	
B.	Priori[illegible]	II.	
C.	Doubl[illegible]	III.	
D.	Simpl[illegible]	IV.	

Choose the c[illegible]ns given below:
1. A-I, B-III[illegible]
2. A-II, B-II[illegible]
3. A-II, B-II[illegible]
4. A-IV, B-I[illegible]
Answer:
C
Sol:
In this ques[illegible] the correct explanation [illegible]
A. Circular Q[illegible]ws elements to be added [illegible] scheduling, where proc[illegible]anaged in a round-robin fashion, ensuring that each process gets a fair share of the CPU's time.
B. Priority Queue is used in III. Dijkstra Algorithm. A priority queue allows us to efficiently retrieve the element with the highest priority. In the Dijkstra algorithm, this data structure is critical for selecting the next node to process based on the shortest distance, which makes it suitable for finding the shortest path in a graph.
C. Double Ended Queue (Deque) is used in IV. Palindrome Checking. A deque allows access to both ends of the queue, which is ideal for palindrome checking because it allows comparing the first and last characters, then moving inward until all characters are checked for symmetry.
D. Simple Queue is used in I. Print Queue. A simple queue operates in a First-In-First-Out (FIFO) manner, which fits the needs of a print queue. As documents arrive in the print queue, the first document to arrive is the first one to be printed.
Information Booster:

1. Circular Queue in CPU Scheduling: A circular queue in CPU scheduling ensures that all processes are given equal opportunity to use the CPU by cycling through them repeatedly. This avoids starvation where some processes may never get scheduled. The circular nature of the queue allows for efficient memory usage and guarantees that each process gets a fair turn.
2. Priority Queue in Dijkstra's Algorithm: Priority queues play an essential role in algorithms like Dijkstra's algorithm, which finds the shortest path in a graph. In this case, the priority queue helps select the next node with the smallest tentative distance, making the algorithm efficient. The priority queue is implemented using a heap to ensure that the minimum distance node is accessed in logarithmic time, leading to an overall efficient algorithm.
3. Deque in Palindrome Checking: A deque (Double-Ended Queue) is perfect for palindrome checking because it allows access to both the front and the back of the sequence. By comparing the characters at both ends and removing them from the deque one by one, you can efficiently check if the string reads the same for
4. Simple Qu
is the perfec
printed. Thi

Q74. What i
n?
1. O(1)
2. O(n)
3. O(log n)
4. O(n log n)
Answer:
C
Sol:
A tail recurs
iterative ste
complexity
Information
1. Tail recur
2. Balanced
3. Auxiliary
4. Quick Sor
Additional K
1. Option (a)
2. Option (b
3. Option (d) O(n log n) This denotes average time complexity, not space usage.

Q75. A relation R on a set A is called a partial order if it is:
1. Reflexive, Symmetric and Transitive.
2. Reflexive, Antisymmetric and Transitive.
3. Irreflexive and Symmetric.
4. Symmetric and Transitive.
Answer:
B
Sol:
A partial order relation is a binary relation on a set that satisfies three key properties: reflexivity, antisymmetry and transitivity. These properties together ensure that elements can be compared in a

non-linear but logically consistent way. Partial orderings are used to model structures like hierarchies, subsets and task dependencies, where not all elements are necessarily comparable.
Information Booster:
1. Reflexive: Every element is related to itself: aRa.
2. Antisymmetric: If aRb and bRa, then a = b.
3. Transitive: If aRb and bRc, then aRc.
4. Partial Order:
A relation that satisfies all three properties.
Not all pairs need to be comparable.
Denoted as a poset (Partially Ordered Set).

Additional Knowledge:

- **Example of Partial Order:** Subset relation $\subseteq$ on power set P(S)
- Reflexive: A $\subseteq$ A
- Antisymmetric: $A \subseteq B$ and $B \subseteq A => A = B$
- Transitive: $A \subseteq B, B \subseteq C => A \subseteq C$
- **Total Order:** A special kind of partial order where **every pair** of elements is comparable.

Q.76 A partial order $\leq$ is defined on the set $S = \{x, a_1, a_2, \ldots, a_n, y\}$ as $x \leq a_i$ for all i and $a_i \leq y$ for all i, where $n \leq 1$. The number of total orders on the set S which contain the partial order $\leq$ is:

A. n!
B. n
C. n + 2
D. 1

Answer: A

Sol: The partial order specifies that x is less than or equal to every a_i, and each a_i is less than or equal to y. This means the elements $a_1, a_2, \ldots, a_n$ lie between x and y in the order. However, the order among the a_i elements themselves is not specified, so they can be arranged in any order relative to each other. The total number of ways to arrange n elements is n! (factorial of n). Since the partial order constraints $x \leq a_i \leq y$ must be respected, the only freedom is in permuting the a_i elements. Therefore, the number of total orders containing the partial order is n!.

Information Booster:

1. A **partial order** defines ordering constraints but may leave some elements incomparable.
2. A **total order** is a linear order where every pair of elements is comparable.
3. Extending a partial order to total orders involves filling in unspecified relationships.
4. The factorial n! counts permutations of n distinct elements.
5. Constraints from partial order reduce freedom to permute only elements not fixed by order.
6. Partial order extensions are key in order theory and combinatorics.
7. Understanding orders helps in sorting, scheduling, and data structuring.

Additional Knowledge:

- Partial orders can be represented by **Hasse diagrams** to visualize constraints.
- Total orders extending a partial order are also called **linear extensions**.
- Counting linear extensions is generally a complex combinatorial problem.

Q77. A Lex compiler generates _____.

1. Lex object code
2. Transition tables
3. C tokens
4. None of the above

Answer:

B

Sol:

A Lex compiler (or Lexical Analyzer Generator) is a tool used to generate lexical analyzers, which are programs used for tokenizing input strings. When the Lex source code is compiled, the Lex compiler produces transition tables that represent a deterministic finite automaton (DFA). These tables are then used to anal

Information

1. Lex Sourc

Written usi

Defines the

2. Output of

Transition t

Associated a

3. Role of Tr

These tables s fro one haracters.

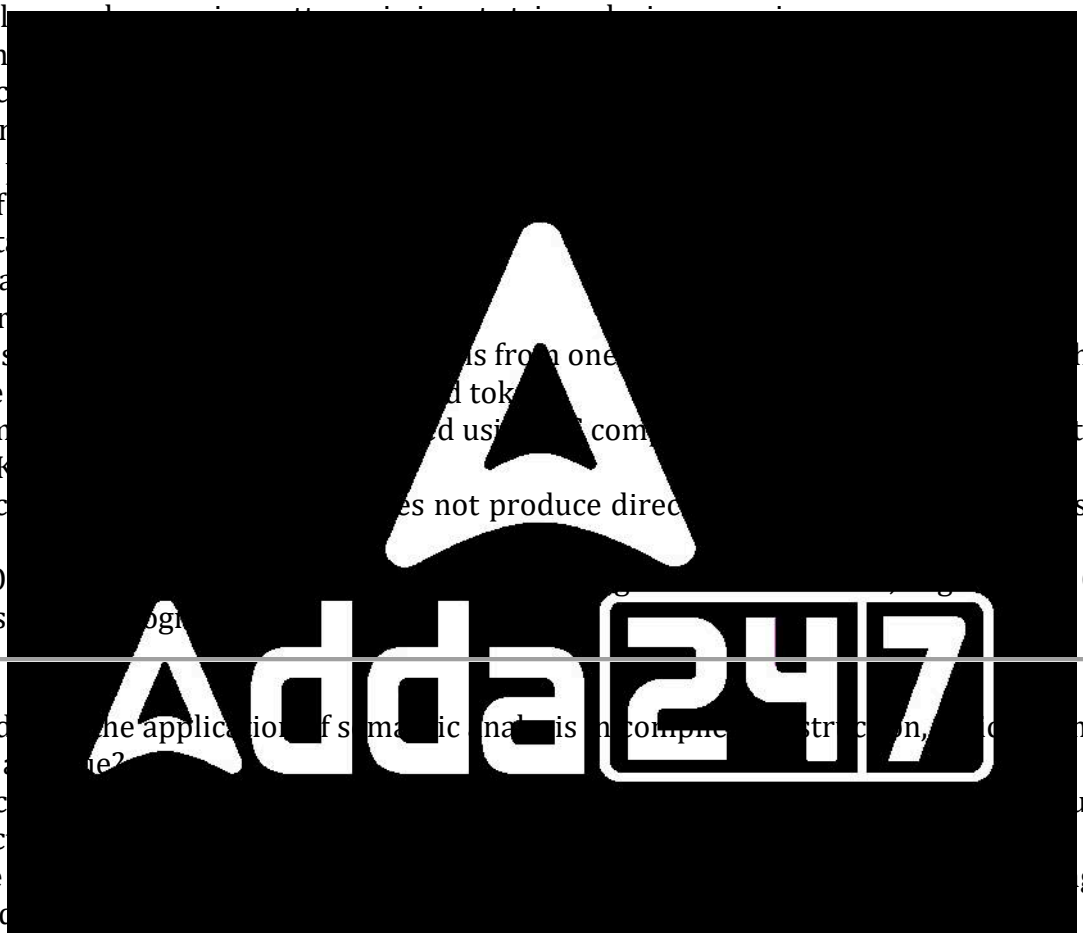
Used by the d tok

4. Final Com d usi com table.

Additional K

Lex object c es not produce direc source code first.

C tokens (O C code that implements



Q78. Consid he applica f s ic anal is compile str n, he following statements a e?

A. Semantic ucts beyond their syntac

B. Attribute g and scope resolution, c

C. Bottom-up parsing techniques are typically employed during the lexical analysis phase of compiler construction to identify tokens and build the parse tree.

D. Semantic analysis is primarily concerned with optimizing the runtime performance of compiled programs.

E. Semantic analysis plays a crucial role in compiler optimization, including dead code elimination and loop unrolling.

Choose the correct answer from the options given below:

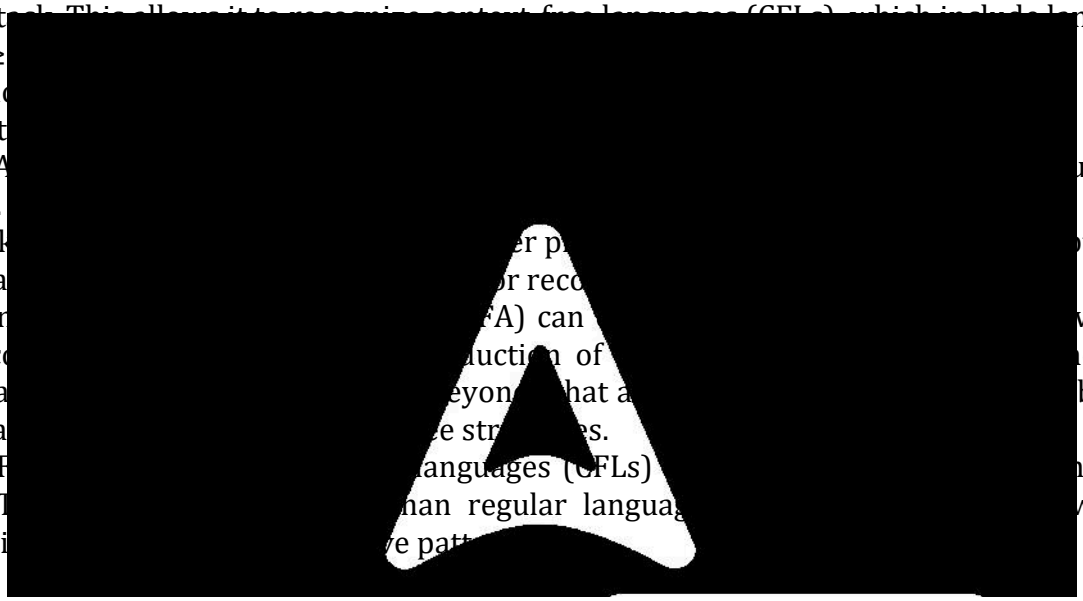
1. A and B only
2. C and D only
3. C and E only
4. B, D, E

Answer:

A
Sol:
Semantic analysis in compiler construction verifies the meaningfulness and consistency of program constructs beyond mere syntax correctness. Attribute grammars play a crucial role in specifying static semantic rules like type checking and scope resolution.
Information Booster:
1. Statement A (Correct): Semantic analysis ensures that syntactically valid constructs also make logical sense, handling aspects like type compatibility, variable binding, and proper declaration. It goes beyond syntax checks by verifying meaningful interpretations of program constructs.
2. Statement B (Correct): Attribute grammars provide a structured method to define static semantics. They associate attributes with grammar symbols and define semantic rules to enforce type checking, scoping rules, and other semantic constraints during compilation.
Additional K
• Statement
analysis, no
automata, n
• Statement
checking, sc
in the later p
• Statement
optimizatio
separate fro
• Semantic
semantics a

Q79. Which
1. A PDA ca
2. A PDA ca
3. A PDA is s
4. PDAs can
1. Only 1, 2
2. Only 1 a
3. Only 2 a
4. Only 1 a
Answer:
A
Sol:
Let's analyze each of the given statements regarding Pushdown Automata (PDAs):
1. A PDA can recognize all regular languages.
Correct: A Pushdown Automaton (PDA) is a more powerful computational model than a Finite Automaton (FA). Since regular languages are a subset of context-free languages, a PDA can indeed recognize all regular languages. This is because regular languages can be recognized by finite automata, and a PDA is capable of simulating a finite automaton (by not using the stack or using it trivially).
2. A PDA can accept a language by final state or empty stack.
Correct: A PDA can accept a language in two different ways:
Final state acceptance: The PDA enters an accepting state when it has finished processing the input.
Empty stack acceptance: The PDA accepts the input when the stack is empty after processing the input, regardless of whether the machine reaches a final state. Both methods are valid ways for a PDA to accept a language.

3. A PDA is strictly more powerful than a finite automaton.
Correct: A PDA is strictly more powerful than a finite automaton because it has access to a stack, allowing it to recognize a broader class of languages — specifically, context-free languages (CFLs). A finite automaton can only recognize regular languages, which are a proper subset of context-free languages.
4. PDAs can be used to recognize non-context-free languages.
Incorrect: A PDA can only recognize context-free languages (CFLs), which are a strict superset of regular languages but a subset of context-sensitive and recursively enumerable languages. PDAs are not capable of recognizing non-context-free languages (such as some context-sensitive or recursively enumerable languages). Therefore, this statement is incorrect.
Information Booster:
1. Pushdown Automata (PDA): A PDA is similar to a finite automaton, but it has the additional capability of using a st
2. Acceptanc
PDAs have t
Final State A
Empty Stack
These two a
3. Comparin
subset of c
recognize la
balanced pa
4. Context-F
automata. T
structures li



Q.80 Consider the following Pushdown Automaton (PDA) with the alphabet $\Sigma = \{a, b\}$ and the stack alphabet $\Gamma = \{Z_0, A\}$. The PDA starts in state q_0, with the stack initially containing Z_0, and it accepts the language $L = \{a^n b^n | n \geq 1\}$. Which of the following is a correct description of how this PDA operates?

A. The PDA pushes an 'A' for every 'a' encountered and pops an 'A' for every 'b' encountered.
B. The PDA pops 'A' for every 'a' encountered and pushes 'A' for every 'b' encountered.
C. The PDA only accepts strings of the form $a^n b^n c^n$, where the number of 'a's, 'b's, and 'c's are equal.
D. The PDA does not use the stack to accept strings of $a^n b^n$.

Answer: A

Sol: We are given a **Pushdown Automaton (PDA)** with:
· **Alphabet** $\Sigma = \{a, b\}$
· **Stack alphabet** $\Gamma = \{Z_0, A\}$, where Z_0 is the initial stack symbol and A is used for pushing and popping from the stack.
The PDA is designed to accept the language $L = \{a^n b^n \mid n \geq 1\}$, which consists of strings where the number of 'a's is equal to the number of 'b's.

How the PDA works?

1. **Initialization**: The PDA starts in state q_0 with the stack containing Z_0.
2. **Processing 'a's**:
- For every 'a' encountered in the input string, the PDA pushes an A onto the stack. This ensures that each 'a' is matched by an A in the stack.
- This is because the number of 'a's needs to be counted, and the PDA uses the stack to keep track of the count.
3. **Processing 'b's**:
- When the PDA encounters a 'b', it pops an A from the stack. This is done to match each 'b' with an earlier 'a' (represented by the A in the stack).
- If the number of 'b's exceeds the number of 'a's, the stack will become empty before all 'b's are processed, causing the machine to reject the string.

4. **Acceptance**: The PDA accepts the string if it reaches the end of the input string with the stack containing only the initial symbol Z_0, which means all the A's pushed for the 'a's have been popped for the 'b's. Thus, the number of 'a's and 'b's must be equal for the string to be accepted.

Now, let's analyze the options:

1. **Option (a) The PDA pushes an 'A' for every 'a' encountered and pops an 'A' for every 'b' encountered**: This is the **correct description** of how the PDA works. The PDA pushes an 'A' for each 'a' and pops an 'A' for each 'b' to ensure the string has the same number of 'a's and 'b's.
2. **Option (b) The PDA pops 'A' for every 'a' encountered and pushes 'A' for every 'b' encountered**: This is **incorrect**. The PDA should push an 'A' for every 'a' and pop an 'A' for every 'b', not the other way around.
3. **Option (c) The PDA only accepts strings of the form $a^nb^nc^n$, where the number of 'a's, 'b's, and 'c's are equal**: This is **incorrect**. The given language is a^nb^n, not $a^nb^nc^n$. The PDA does not deal with 'c's, and it is designed to accept strings with equal numbers of 'a's and 'b's.
4. **Option (d) The PDA does not use the stack to accept strings of a^nb^n**: This is **incorrect**. The stack is essential for the PDA to keep track of the number of 'a's and 'b's. Without the stack, it would not be possible to ensure that the number of 'a's equals the number of 'b's.

Information Booster:

1. **Pushdown Automaton (PDA)**: A PDA is a type of automaton that uses a stack to provide additional memory beyond its finite states. The stack allows the PDA to recognize context-free languages.
2. **Language a^nb^n**: This is a classic example of a context-free language, where the number of 'a's must be equal to the number of 'b's. A PDA can recognize this type of language by using its stack to count and match 'a's with 'b's.
3. **Stack Usage**: The stack in a PDA is used for matching and counting, which is essential for languages like a^nb^n, where the order and count of symbols are important.

Additional Knowledge:

- PDAs can be used to accept a broader class of languages, including context-free languages, which cannot be recognized by finite automata alone. The stack allows for the additional memory needed to handle nested or recursive structures.

Q.81 Let L_D be the set of all languages accepted by a PDA by final state and L_E be the set of all languages accepted by empty stack. Which of the following is true?

A. $L_D = L_E$
B. $L_D \subset L_E$
C. $L_D \supset L_E$
D. None of the above

Answer: A

Sol:
A Pushdown Automaton (PDA) can accept a language using two different acceptance criteria: by reaching a final state or by emptying its stack. Although the mechanisms differ, both methods are equivalent in terms of the class of languages they recognize — the context-free languages (CFLs). Thus, the set of languages accepted by final state and by empty stack are the same.
Information Booster:
1. Pushdown Automaton (PDA): A PDA is a computational model that extends the power of finite automata with a stack, allowing it to recognize context-free languages.
2. Acceptance by Final State:
In this approach, a string is accepted if, after processing the input, the PDA enters a final (accepting) state.
The contents of the stack at the end do not matter.
3. Acceptance by Empty Stack
A string is a
egardless of
the current
The final sta
4. Equivalen
For every P
ts the same
language by
Therefore, t
nethod
context-free
languages (C
5. Formal Th
The equival
ck a
inal s
in automata
theory.
This implies
Additional K
Acceptance
er to
when
context-free
languages.
Acceptance
sta
mo
sed
s and th
re
nalysis
u
ts structural
clarity.
Even thoug
pe
tio
a
differ
, t
y can s
mulate
ach oth
e
be
w
y modifying
transitions
ng dummy sy
ols
The class of
xt f
ng
ages (C
L
s) is e
actly th
set of languages accepted by
using either
acceptance

Q82. Match

	List – I		
A.	Lexical analysis	I.	Finite automation
B.	Code optimization	II.	DAG's
C.	Code generation	III.	Syntax trees
D.	Parsing programming	IV.	Push down automation

1. A – I, B – II, C – III, D – IV
2. A – II, B – III, C – IV, D – I
3. A – IV, B – I, C – II, D – III
4. A – II, B – IV, C – I, D – III

Answer:
A
Sol:
To solve this question, we match each compiler phase with the most relevant concept or data structure:

1. Lexical Analysis → Finite Automation (I)
Lexical analyzers use Finite Automata (typically DFA) to recognize patterns (tokens) in source code.
2. Code Optimization → DAG's (II)
Directed Acyclic Graphs (DAGs) are used for optimizing basic blocks of code by identifying common subexpressions.
3. Code Generation → Syntax Trees (III)
Code is generated using Syntax Trees which represent the syntactic structure of the program and guide how instructions are produced.
4. Parsing Programming → Push Down Automation (IV)
Parsers (especially for context-free grammars) use Push Down Automata which employ a stack for handling nested structures.
Information Booster:
1. Finite Au keywords, identifiers).
2. DAG (Dire ion.
3. Syntax Tr
4. Push Dow for context-free gramma

Q83. Match

	List – I (Grami	II oduction ru
A.	Choms	→ AB → a, B
B.	Greiba	→ aS
C.	S-gram	S → AS | a, A → SA
D.	LL gra	S → bSS | aS | c

Match the c
1. A - i, B -
2. A - ii, B - ii
3. A - iii, B - iv
4. A - i, B - iv
Answer:
A
Sol:
Information
- Different
- Productio
- We must match each grammar with its valid standard form.

Concept / Definitions Used
1. Chomsky Normal Form (CNF)
A grammar is in CNF if every production is of the form:
- A → BC (two non-terminals), or
- A → a (single terminal)

Rule (i):
S → AB
A → a
B → b
→ Strictly follows CNF rules.
A → i

2. Greibach Normal Form (GNF)
A grammar is in GNF if every production starts with a terminal symbol, followed by zero or more non-terminals.
Rule (ii):
S → aSb | ab
→ Starts with terminal a.
B → ii
3. S-Grammar
An S-grammar generally:
- Allows left recursion
- Productions often start with non-terminals

Rule (iii):
S → AS | a
A → SA | b
C → iii
4. LL Gramm
An LL grami
- Has no le
- Is suitab

Rule (iv):
S → bSS | aS
→ No left re
D → iv
Final Conclu
A-i, B-ii, C-ii

Q84. Which tion of table?
(A) Storage
(B) Checkin pa
(C) Suppres icate r mess ges.
Choose the opt n:
1. (A) and y
2. (A) and v
3. (B) and
4. (A), (B) a

Answer:
D
Sol:
A symbol table is a data structure used in programming languages and compilers to store information about variables, functions, objects, and other entities in the source code. It has multiple applications throughout the different phases of compilation and interpretation. In particular, the symbol table helps in tasks such as storage allocation, checking type compatibility, and suppressing duplicate error messages.
Information Booster:
1. Storage Allocation: The symbol table is used to store information about variables such as their names, types, and memory locations. During compilation or execution, the symbol table helps in assigning memory locations to variables for proper storage allocation.

2. Checking Type Compatibility: The symbol table is crucial for type checking during compilation. It helps ensure that the operations on variables are valid according to their declared types (e.g., ensuring that a string is not added to an integer).
3. Suppressing Duplicate Error Messages: A symbol table helps to track variables, functions, and objects in the program. By maintaining information about previously declared identifiers, it can suppress the re-declaration error messages, ensuring that each identifier is reported only once.
4. Symbol Resolution: It is also essential in resolving symbols used in a program, such as distinguishing between variables and functions with the same name but different scopes.
5. Semantic Analysis: Symbol tables support the semantic analysis phase of a compiler, helping ensure the program follows the rules of the language being compiled.

Q.85 For $L_k = \{ w \in \{a,b\}^* \mid$ the k-th symbol from the right is $a \}$ with $k \geq 1$. The minimum number of DFA states required is:

A. k
B. k + 1
C. 2^k
D. k + 2

Answer: C

Sol: To decide whether the **k-th symbol from the right** is 'a' while scanning left → right, a DFA must know the **last k symbols** seen so far at every step. That's because the final decision after the input ends depends exactly on that k-length suffix. There are 2^k possible k-length strings over {a, b}; by the **Myhill–Nerode** theorem, each such suffix is pairwise distinguishable (for two different suffixes, there exists some continuation that makes one accepted and the other rejected). Hence, no two of these suffix-histories can be merged, so the minimal DFA needs 2^k states. (Checks: for k = 1 "ends with a" → 2 states; for k = 2 "second last is a" → 4 states.)

Information Booster:

1. **Core idea:** The acceptance depends solely on the **k-length suffix** of the input at end-of-file.
2. **State meaning:** Each state represents one of the 2^k possible binary strings over {a, b} of length k (a shift register of the last k symbols).
3. **Transitions:** On input symbol $x \in \{a,b\}$, shift left and append x; this is a deterministic mapping among the 2^k states.
4. **Accepting states:** Those whose stored k-suffix has its **first (leftmost) symbol** = **a** (i.e., the k-th from right of the full input is a).
5. **Myhill–Nerode argument:** Different k-suffixes are distinguishable by choosing a continuation that fills to end so that the k-th-from-right differs, proving the 2^k lower bound.
6. **Tightness:** The constructed "shift-register" DFA attains 2^k states, matching the lower bound → **minimal**.
7. **Small-k sanity checks:** $k = 1 \Rightarrow 2$ states; $k = 2 \Rightarrow 4$; $k = 3 \Rightarrow 8$; matches 2^k.

Q86. Match the following:

	List – I (Disk Scheduling Algorithms)		List – II (Description)
A.	FCFS	I.	Requests are processed in the order they arrive, regardless of position on the disk.
B.	SSTF	II.	The disk arm moves to the request closest to the current position.
C.	SCAN	III.	The disk arm moves in one direction, servicing requests until the end is reached, then reverses direction.
D.	C-SCAN	IV.	The disk arm moves in one direction only, wrapping around once it reaches the end.

1. A → I, B → II, C → III, D → IV
2. A → II, B → I, C → IV, D → III
3. A → III, B → IV, C → II, D → I
4. A → IV, B → III, C → I, D → II

Answer:

A

Sol:

Here's the correct match between the disk scheduling algorithms and their descriptions:

1. FCFS (A): I - First Come First Served (FCFS) is the simplest disk scheduling algorithm. Requests are processed in the order they arrive, regardless of their position on the disk. This can lead to inefficiencies as the disk arm may travel a lot of unnecessary distance.

2. SSTF (B): II - Shortest Seek Time First (SSTF) is a more efficient scheduling algorithm. In SSTF, the disk arm mo[illegible] time for each request.

3. SCAN (C) [illegible] oving in one direction, se[illegible] ng direction to service th[illegible]

4. C-SCAN ([illegible] ection when the end of [illegible] ound to the beginning o[illegible] more [illegible] o SCAN.

Information[illegible]

1. FCFS: The [illegible] implement[illegible] e because it does not mi[illegible]

2. SSTF: Wh[illegible] e for [illegible] requ[illegible] ere requests far from the[illegible] e serviced.

3. SCAN: Th[illegible] e total seek time co[illegible] e inefficient when there [illegible] of the [illegible]

4. C-SCAN: C[illegible] rticularly in systems wh[illegible] ts [illegible] di[illegible] te[illegible] disk.

Q.87 Using priority scheduling algorithm, find the average waiting time for the following set of processes given with their priorities in the order, Process, Burst Time, Priority, respectively.

- $P_1 : 10 : 3$
- $P_2 : 1 : 1$
- $P_3 : 1 : 1$
- $P_4 : 1 : 5$
- $P_5 : 5 : 2$

A. 8 ms
B. 5.4 ms
C. 7.75 ms
D. 3 ms

Answer: B

Sol: The average waiting time is calculated using non-preemptive **priority scheduling**, where the process with the **lowest priority number** is scheduled first.

Given Processes (Burst Time : Priority):

- $P_1 : 10 : 3$
- $P_2 : 1 : 1$
- $P_3 : 1 : 1$
- $P_4 : 1 : 5$
- $P_5 : 5 : 2$

Sort by Priority (ascending):

- P_2 and P_3: Priority 1 (tie — assume FCFS)
- P_5 : Priority 2
- P_1 : Priority 3
- P_4 : Priority 5

Execution Order (by priority and FCFS):

- $P_2 \rightarrow P_3 \rightarrow P_5 \rightarrow P_1 \rightarrow P_4$

Completion Times:

- P_2 : Starts at 0 → Completes at 1 → Waiting Time = 0
- P_3 : Starts at 1 → Completes at 2 → Waiting Time = 1
- P_5 : Starts at 2 → Completes at 7 → Waiting Time = 2
- P_1 : Starts at 7 → Completes at 17 → Waiting Time = 7
- P_4 : Starts at 17 → Completes at 18 → Waiting Time = 17

Waiting Times:

$P_2 = 0, P_3 = 1, P_5 = 2, P_1 = 7, P_4 = 17$

Average Waiting Time:

$$\frac{0+1+2+7+17}{5} = \frac{27}{5} = 5.4ms$$

Q88. Match

	List – I		List
A.	Handsh	I.	I/ interfa e mo s GPU tha de ired sfer.
B.	Progra /O	II	R qu s t control signals orking pp e tion.
C.	Interru ated I/O	III	H s l al m er y ontrol et f O
D.	I/O pr r	IV	R uired PU to ch k the I/O flag & perform tra

Choose the
Match the co
1. A-I, B-II,
2. A-II, B-IV,
3. A-II, B-IV, C-I, D-III
4. A-IV, B-III, C-II, D-I

Answer:

C

Sol:

Let's match List-I with List-II according to the descriptions of I/O types.

Given Lists:

List-I:

(A) Handshaking

(B) Programmed I/O

(C) Interrupt-initiated I/O

(d) I/O Processor

List-II:
(I) I/O interface informs the CPU that the device is ready for transfer.
(II) Requires two control signals working in opposite directions.
(III) Has local memory and control large set of I/O devices.
(IV) Requires CPU to check the I/O flag & perform transfer.
Matching:
(A) Handshaking
• Handshaking involves signaling between the CPU and peripheral devices to indicate readiness for data transfer.
• It typically requires two control signals working in opposite directions to complete the communication process.
• Thus, (A) matches with (II).
(B) Program
• Programm ally without interrupts.
• Thus, (B)
(C) Interrup
• In this met
• Thus, (C)
(d) I/O Proc
• I/O proce y and c I/O devices autonomous
• Thus, (d)
Final Answe
The correct (C)-(I), (d)-(III).

Q89. An op ble fo reso emory, and input/outpu sses can be allocated to fo exe a me.
1. 4
2. 3
3. 2
4. 5

Answer:
B
Sol:
With 3 CPU n time. The remaining processes will have to wait in the ready queue for an available CPU core. This highlights the importance of parallel processing and how the number of cores limits the concurrent execution of processes.
Information Booster:
1. CPU Cores – A CPU core is a single processing unit capable of executing tasks. More cores allow for better parallel processing.
2. Process Allocation – The OS assigns processes to available CPU cores, ensuring that each core handles a separate task in multitasking systems.
3. Queueing – Processes that cannot be assigned to a core immediately are placed in the ready queue, where they wait until a core becomes available.
4. Multitasking – Operating systems use multitasking to handle multiple processes at the same time, maximizing the use of available CPU cores.

Additional Knowledge:
Multithreading – A technique where multiple threads of a process can be executed concurrently.
Load Balancing – The OS may also implement load balancing to distribute processes evenly across available CPU cores.
Context Switching – When processes need to share CPU time, the OS performs context switching to swap between processes.

Q90. A disk has cylinders numbered 0 to 199. The disk head is currently at cylinder 90 and is moving towards lower-numbered cylinders. The pending request queue (in order of arrival) is:
10, 130, 50, 160, 20, 70
If the disk uses the SCAN (elevator) scheduling algorithm, what is the total head movement (in cylinders) needed to serve all requests?

1. 150
2. 170
3. 190
4. 250

Answer:
D
Sol: The SCA dire p to the end of the disk, a
Cylinders Ra
Current Pos
Initial Direc cylin (0).
Request Que
Step 1: Mov ers (0)
The request cylinder number, a d continues until it reach der 0)

1. **Serve 70:** $90 \rightarrow 70$. Movement: $|90 - 70| = 20$
2. **Serve 50:** $70 \rightarrow 50$. Movement: $|70 - 50| = 20$
3. **Serve 20:** $50 \rightarrow 20$. Movement: $|50 - 20| = 30$
4. **Serve 10:** $20 \rightarrow 10$. Movement: $|20 - 10| = 10$
5. **Move to End (0):** $10 \rightarrow 0$. Movement: $|10 - 0| = 10$

· **Total Movement Down:** $20 + 20 + 30 + 10 + 10 = \mathbf{90}$

Step 2: Reversal and Movement Towards Higher Cylinders (199)

The head now reverses direction from cylinder 0. The remaining requests, sorted by cylinder number, are 130 and 160.

1. **Serve 130:** $0 \rightarrow 130$. Movement: $|0 - 130| = 130$
2. **Serve 160:** $130 \rightarrow 160$. Movement: $|130 - 160| = 30$

· **Total Movement Up:** $130 + 30 = \mathbf{160}$

Total Head Movement

$$\text{Total Movement} = \text{Movement Down} + \text{Movement Up}$$
$$\text{Total Movement} = 90 + 160 = 250$$

Information Booster

The **SCAN** (Elevator) algorithm is a disk scheduling strategy designed to reduce head movement and improve system throughput.

1. **Mechanism:** The head continuously sweeps back and forth across the entire disk, serving requests in its path. It moves in one direction until it reaches the last cylinder (or the farthest request in that direction) and then reverses.

2. **Fairness:** SCAN provides better fairness than SSTF (Shortest Seek Time First) because it prevents starvation; requests waiting at the far end of the disk will eventually be served when the head reverses direction and sweeps toward them.
3. **Latency:** Requests just missed by the head in one direction will have to wait for almost a full sweep, leading to high variance in response time.
4. **C-SCAN (Circular SCAN):** A variant that only sweeps in one direction (e.g., from 0 to 199), and then jumps back to 0 without servicing any requests on the return sweep. This ensures that new requests arriving near 0 do not have to wait as long for service as they would in the standard SCAN method.

Additional Knowledge

· **Head Movement Calculation:** The key to all disk scheduling problems is calculating the **absolute difference** between the current cylinder position and the next cylinder position for every move and summing these differences.

· **Other Scheduling Algorithms:**

· **FCFS (First Come, First Served):** Simplest, but poor performance. Total movement depends solely on the request arrival order.

· **SSTF (Shortest Seek Time First):** Always chooses the request closest to the current head position. High performance, but can cause **starvation** for requests at the edges.

· **LOOK/C-LOOK:** Variations of SCAN/C-SCAN that reverse direction (LOOK) or jump (C-LOOK) immediately upon reaching the farthest request in the current direction, instead of sweeping all the way to the physical end of the disk. These are generally more efficient than their non-LOOK counterparts.

Q91. What i n & Swap out) in a sy th a transfer rate of 30 M nvolved.

1. 1.024 se
2. 1.00 sec
3. 0.512 se
4. 12 sec

Answer:

A

Sol:

To calculate , we need to consider the following parameters:

Transfer rate = 30 MBps

Average latency = 12 ms = 0.012 seconds

Process size = 15 MB

We need to calculate the total swap time, which includes both swap in and swap out operations.

Step-by-Step Calculation:

1. Total Data: Since we are dealing with swap in and swap out, the total data transferred is twice the process size:

$$\text{Total data} = 2 \times 15\,\text{MB} = 30\,\text{MB}$$

2. **Transfer Time**: The transfer time is calculated using the formula:

$$\text{Transfer Time} = \frac{\text{Total Data}}{\text{Transfer Rate}}$$

Given:

- Total Data = 30 MB
- Transfer Rate = 30 MBps

$$\text{Transfer Time} = \frac{30\,\text{MB}}{30\,\text{MBps}} = 1\,\text{second}$$

3. **Average Latency**: The average latency is given as **12 ms** (milliseconds), which is equivalent to:

$$\text{Latency} = 12\,\text{ms} = 0.012\,\text{seconds}$$

Since, latency affects both the **swap in** and **swap out** operations, we need to account for it twice:

$$\text{Total Latency} = 2 \times 0.012\,\text{seconds} = 0.024\,\text{seconds}$$

4. **Total Swap Time**: The total swap time is the sum of the **transfer time** and the **total latency**:

$$\text{Total Swap Time} = \text{Transfer Time} + \text{Total Latency}$$

$$\text{Total Swap Time} = 1\,\text{second} + 0.024\,\text{seconds} = 1.024\,\text{seconds}$$

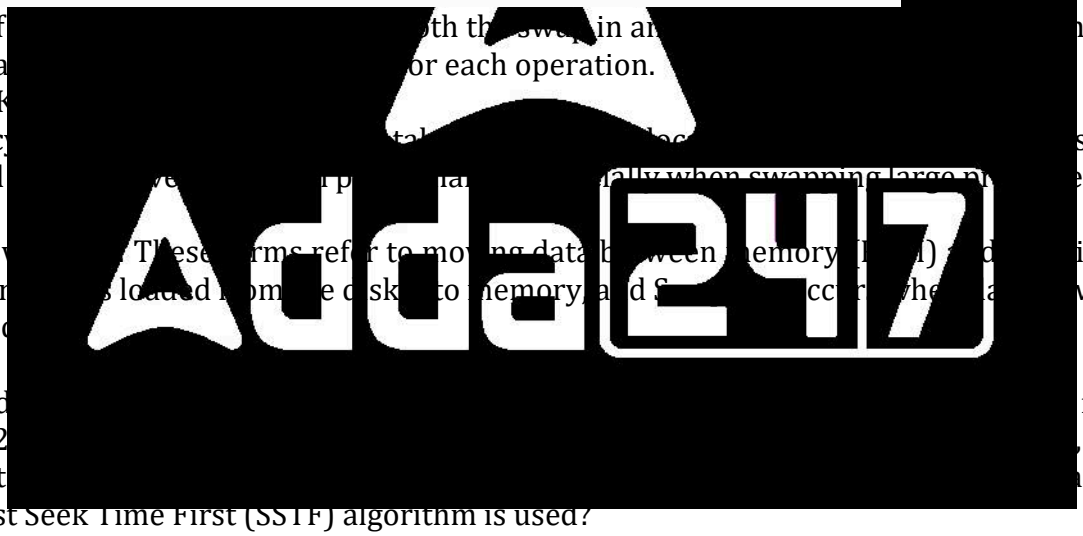
Conclusion:

The **total swap time** for the given process is **1.024 seconds**.

Information Booster:

1. **Swap Time**: **Swap time** refers to the time taken to transfer a process from the memory to disk (swap out) or from disk to memory (swap in). It is influenced by the **data transfer rate** and **latency** of the storage system.
2. **Transfer Rate**: The **transfer rate** is the speed at which data can be moved between memory and disk. In this case, it's 30 MBps (megabytes per second), which determines how long it will take to transfer 30 MB of data.
3. **Latency**: **Latency** refers to the time delay before a data transfer begins. Even though the transfer rate is high, there is a small delay associated with the request to transfer data.

4. Impact of [illegible] operations occur, the la[illegible] or each operation.
Additional K[illegible]
Disk Latency[illegible] made. It is an essential [illegible] in and out of memory.
Swap In / S[illegible] These terms refer to moving data between memory [illegible] disk. Swap In occurs when [illegible] written from memory to [illegible]



Q92. Consid[illegible] for cylinder 10, 22, 20, 2[illegible], what is the time taken t[illegible] adjacent one and Shortest Seek Time First (SSTF) algorithm is used?

1. 240 milliseconds
2. 96 milliseconds
3. 120 milliseconds
4. 112 milliseconds

Answer:

C

Sol: We are tasked with calculating the time it takes to satisfy all disk requests using the Shortest Seek Time First (SSTF) algorithm. This algorithm selects the request that is closest to the current position of the disk head. The Shortest Seek Time First (SSTF) algorithm selects the disk request that is closest to the current head position. It minimizes the seek time by selecting the nearest cylinder in each step.

Given:

• Total number of cylinders: 60
• Disk requests: 10, 22, 20, 2, 40, 6, 38 (in the given order)
• Initial head position: Cylinder 20
• Time to move from one cylinder to the next: 2 milliseconds
• Disk algorithm: SSTF (Shortest Seek Time First)
Step-by-Step Solution:
We start at cylinder 20 and use the SSTF algorithm to pick the nearest request from the current head position.
Sequence of Processing:
1. Start at cylinder 20:
o Requests: 10, 22, 20, 2, 40, 6, 38
o The current head is already at 20. No move required.
o Seek time: 0 ms
2. Next close
o Requests:
o The neare
o Seek time:
3. Next close
o Requests:
o The neare ssed),
o Seek time: s = 24 m
4. Next close
o Requests:
o The neare
o Seek time: ms = 8 ms
5. Next close
o Requests:
o The neare
o Seek time: cy 4 * = 8
6. Next close er to
o Requests:
o The neare est to 2 is 8.
o Seek time: 8 ders → 6 * 2 m = 72 m
7. Next close
o Requests:
o The neare
o Seek time:
Total Time:
• Total seek time = 4 + 24 + 8 + 8 + 72 + 4 = 120 milliseconds
Correct Answer: 120 milliseconds
Information Booster:
1. SSTF (Shortest Seek Time First) minimizes the time to move between cylinders by always selecting the closest request to the current head.
2. Seek Time: The time to move between two cylinders is calculated as the absolute difference between their positions multiplied by the time to move from one cylinder to the next (2 ms in this case).
3. The total seek time is the sum of the individual seek times calculated for each request.
Additional Knowledge:
• Disk Scheduling algorithms like SSTF are designed to optimize seek time in hard disk drives, leading to more efficient data retrieval.

• Starvation: In SSTF, requests located far away from the current head can experience starvation, as the head may repeatedly choose closer requests.

Q93. Which Linux command is used to delete files or directories?
1. rm
2. del
3. erase
4. remove

Answer:
A
Sol:
The rm command in Linux is used to delete files or directories. It is a standard command-line utility that removes file
Information
1. The rm c rsive flag is used.
2. Using rm not be easily recovered.
3. Example:
To delete a f
To delete a d
Additional K
Option (b): ws, n inux,
Option (c): e Linu eleti
Option (d): mmand; the correc

Q94. Arrang ation:
1. OS issues
2. Disk head th ra ek)
3. Controller rites a
4. Rotationa p sitio s se or and he a
5. Data is tra ed to mem ory

1. 1 → 2 →
2. 1 → 4 →
3. 1 → 2 →
4. 1 → 3 →

Answer:
A
Sol:
The disk I/O operation typically involves the following sequence of steps:
(1) OS issues command to disk controller:
The Operating System (OS) sends a command to the disk controller, instructing it to perform a read or write operation on a specified location.
(2) Disk head moves to the desired track (seek):
The disk head then moves to the appropriate track that contains the data to be read or written. This is the seek operation, which involves physically moving the head over the spinning disk surface.
(4) Rotational latency positions sector under head:

After the head has reached the correct track, rotational latency comes into play. The disk needs to rotate so that the desired sector is positioned under the disk head. The amount of time this takes depends on the disk speed (measured in RPM).
(3) Controller reads/writes data:
Once the desired sector is under the head, the disk controller proceeds with reading or writing the data from/to the disk surface.
(5) Data is transferred to memory:
Finally, the data is transferred from the disk to the main memory (RAM), completing the I/O operation.
Information Booster:
1. Disk I/O Operation:
Seek Time: Time taken for the disk head to move to the correct track. It depends on the distance the head needs to travel.
Rotational L
the read/wr
Data Transf
2. Disk Perfo
Disk perfor
Seek time (h
Rotational la
Data transfe

Q95. You a

Item			
1			
2			
3			

The knapsac
maximum va
1. 90
2. 110
3. 140
4. 150

Answer:
B

Sol: To solve this problem using the **greedy approach** based on the **value-to-weight ratio**, we need to follow these steps:

1. **Calculate the value-to-weight ratio for each item:**

The **value-to-weight ratio** is calculated as:

$$\text{Value-to-weight ratio} = \frac{\text{Value}}{\text{Weight}}$$

Let's calculate the ratio for each item:

- **Item 1**:

$$\frac{40}{3} \approx 13.333$$

- **Item 2**:

$$\frac{50}{4} = 12.5$$

- **Item 3**:

$$\frac{70}{5} = 14$$

2. **Sort the items by their value-to-weight ratio in descending order:**

The sorted items based on value-to-weight ratio are:

- **Item 3**: 14
- **Item 1**: 13.33
- **Item 2**: 12.5

3. **Start adding items to the knapsack, starting with the item with the highest value-to-weight ratio:**

- The knapsack has a **capacity of 8** units.
- Start with **Item 3** (value 70, weight 5). Adding this to the knapsack leaves **8 - 5 = 3** units of capacity remaining.
- Now, the remaining capacity is **3 units**.

The next ite[illegible] w[illegible]3). [illegible] y of 3 units. Adding this [illegible]apsa[illegible]es a va[illegible]e of 40

4. Stop addi[illegible] o[illegible]e t[illegible]kn[illegible]sack is full [illegible] [illegible]ll ite[illegible]s are con[illegible]dered:

After adding [illegible]1 (value 4[illegible], w[illegible]ght 3), [illegible]e kna[illegible]ack is ful[illegible] (co[illegible] [illegible]igh[illegible]5 + [illegible]= [illegible]ts). No more items can be [illegible]d.

5. Calculate

Total value =

Q96. Which

1. It applies to all regular language.
2. It applies only to infinite regular languages.
3. It applies to all context-free languages.
4. It applies to all recursively enumerable languages.

Answer:

A

Sol:

The Pumping Lemma for regular languages is a property that all regular languages must satisfy. It provides a way to prove that certain languages are not regular by showing that they do not satisfy the conditions of the lemma.

According to the Pumping Lemma for regular languages, for any regular language, there exists a constant p (the pumping length) such that any string s in the language of length at least p can be split into three parts, s = xyz, where:
1. x is a prefix.
2. y is the part that can be repeated (pumped).
3. z is the suffix.
This pumping operation (repeating y any number of times) results in strings that are still within the language. The key idea is that y can be "pumped" (repeated) as many times as needed, and the resulting string will remain in the language.
Information Booster:
1. Pumping Lemma is a tool to prove whether a language is regular or not by checking if it can be "pumped".
2. If a langua
3. The pum ed by finite automata.
Additional K
• It applies o to all regular languages, n
• It applies mp is different from the on
• It applies nguages apply to all recursively cally pro

Q.97 Which language is **not** context-free, best justified via the **pumping lemma for CFLs** or closure arguments?

A. $\{a^i b^i c^j \mid i, j \geq 0\}$
B. $\{a^i b^j c^j \mid i, j \geq 0\}$
C. $\{a^n b^n c^n \mid n \geq 0\}$
D. $\{a^i b^j c^k \mid i, j, k \geq 0\}$

Answer: C

Sol: The language $\{a^n b^n c^n\}$ is **not context-free**. Intuitively, a single stack (as in a PDA) can match **two** dependent counts (e.g., $a^n b^n$) but cannot simultaneously enforce a **third** synchronized count c^n. A standard proof uses the **pumping lemma for CFLs**: let the pumping length be p and take $s = a^p b^p c^p$. Any decomposition $s = uvxyz$ with $|vxy| \leq p$ and $|vy| > 0$ keeps vxy within at most two adjacent blocks among the a's, b's, c's. Pumping v and y changes counts in **only one or two** blocks, breaking the equality $\#a = \#b = \#c$. Hence the language cannot be CFL. In contrast, the other options are CFL (indeed, (d) is regular), so the only non-CFL choice is (c).

Information Booster:

1. **Pumping-lemma intuition (CFLs):** Any sufficiently long string in a CFL can be "pumped" by duplicating/deleting certain substrings while staying in the language; $\{a^n b^n c^n\}$ fails this because all three counters must stay equal.
2. **Two vs. three counters:** A PDA's single stack can synchronize **two** segments (e.g., $a^n b^n$) but not three ($a^n b^n c^n$). Recognizing $\{a^n b^n c^n\}$ requires **at least two stacks** (i.e., Turing power).
3. **Closure aids:** CFLs are closed under **union**, **concatenation**, **Kleene star** and **intersection with regular** languages; they are **not** closed under complement nor intersection in general.
4. **Why (a) is CFL:** $\{a^i b^i c^j\} = \{a^i b^i\} \cdot \{c^j\}$; concatenation of a CFL with a **regular** language is CFL.
5. **Why (b) is CFL:** $\{a^i b^j c^j\} = \{a^*\} \cdot \{b^j c^j\}$; again regular. CFL is CFL.

6. **Why (d) is CFL (indeed regular):** $\{a^i b^j c^k\} = a^* b^* c^*$, a simple regular expression—hence regular $\subset CFL$.

7. **Alternative non-CFL proof:** Assume $\{a^n b^n c^n\}$ were CFL. Intersect it with the regular language $a^* b^* c^*$ (which changes nothing) and apply pumping; contradiction persists, confirming non-context-freeness.

Additional Knowledge (Other Options):

· **(a)** $\{a^i b^i c^j\}$: CFL by **concatenation** of $\{a^i b^i\}$ (CFL) with c^* (regular).

· **(b)** $\{a^i b^j c^j\}$: CFL by **concatenation** $a^* \cdot \{b^j c^j\}$ (CFL).

· **(d)** $\{a^i b^j c^k\}$: **Regular** $(a^* b^* c^*)$; all regular languages are CFLs.

Q.98 Consider the recurrence $T(n) = 3T(n/4) + n^2$. Using the master theorem, what is the time complexity of this recurrence?

A. $O(n^2)$
B. $O(n^3)$
C. $O(n^{\log_4 3})$
D. $O(n \log n)$

Answer: A

Sol: We are given the recurrence:

$T(n) = 3T(n/4) + n^2$

This is a divide-and-conquer recurrence, and we can solve it using the **Master Theorem**.

The general form of a recurrence for divide-and-conquer problems is:

T(n) = aT(n/b) + f(n)

Where:

· a is the number of subproblems,

· b is the factor by which the problem size is reduced in each subproblem,

· f(n) is the cost of dividing the problem and combining the results from the subproblems.

Step 1: Identify values from the recurrence

For our recurrence $T(n) = 3T(n/4) + n^2$, we identify:

· a = 3 (number of subproblems),

· b = 4 (problem size is divided by 4),

· $f(n) = n^2$ (the work done outside the recursive calls).

Step 2: Apply the Master Theorem

The Master Theorem provides a way to determine the asymptotic behavior of divide-and-conquer recurrences. We need to compare the function f(n) to $n^{\log_b a}$, where:

· a = 3,

· b = 4

We compute:

$\log_b a = \log_4 3$

Now we compare the function $f(n) = n^2$ with $n^{\log_4 3}$:

1. **Case 1**: If f(n) is polynomially smaller than $n^{\log_b a}$, i.e., $f(n) = O(n^c)$ where $c < \log_b a$, then $T(n) = O(n^{\log_b a})$.
2. **Case 2**: If f(n) is equal to $n^{\log_b a}$, i.e., $f(n) = \Theta(n^{\log_b a})$, then $T(n) = O(n^{\log_b a} \log n)$.
3. **Case 3**: If f(n) is polynomially larger than $n^{\log_b a}$, i.e., $f(n) = \Omega(n^c)$ where $c > \log_b a$, and regularity conditions are met, then T(n) = O(f(n)).

Step 3: Compare n^2 with $n^{\log_4 3}$

· We already know that n^2 is **larger** than $n^{\log_4 3}$ because $\log_4 3 \approx 0.792$, and $2 > 0.792$.

· Therefore, f(n) = n^2 is polynomially larger than $n^{\log_4 3}$, which corresponds to **Case 3**.

Step 4: Conclusion

Since the function $f(n) = n^2$ is larger than $n^{\log_4 3}$, the time complexity is determined by f(n), and hence:

$T(n) = O(n^2)$

Information Booster:

1. **Master Theorem**: The Master Theorem is a powerful tool for solving divide-and-conquer recurrences. It categorizes the problem based on the comparison between f(n) and $n^{\log_b a}$.
2. **Case 1**: If f(n) is smaller than $n^{\log_b a}$, the solution will be dominated by the recursive calls and will be $O(n^{\log_b a})$.
3. **Case 2**: If f(n) is the same as $n^{\log_b a}$, the solution will include a logarithmic factor, resulting in $O(n^{\log_b a} \log n)$.
4. **Case 3**: If f(n) is larger than $n^{\log_b a}$, the solution will be dominated by f(n), resulting in O(f(n)).

Additional Knowledge:

Understanding the **Master Theorem** is essential for quickly solving recurrences that arise in **divide-and-conquer algorithms**, like **merge sort**, **quick sort** and others.

Q99. Consid ree o er 4, w ere eac node contain maxim of 3 a minimum of 2 keys. If to ore 00 ys in t B+ and ssumin all nod e p fe alanced and utilized, wh e maximu h ht f t tre?

1. 3
2. 4
3. 5
4. 6

Answer:

D

Sol:

To determine the maximum height of a B+ tree given its order and the number of keys, we calculate the number of keys each node can accommodate and then compute the tree height by considering the tree structure from leaves to the root. Given a perfectly balanced B+ tree of order 4 (maximum of 3 keys per node), storing 300 keys requires computing the height step-by-step from leaf nodes to the root, resulting in a maximum height of 6.

Information Booster:

1. Key properties of a B+ tree (Order 4):

o Each internal node has at most 3 keys and 4 pointers.

o Each internal node has at least 2 keys (minimum) and 3 pointers.

o Leaf nodes contain actual data keys. Each leaf node can contain 3 keys at most.

2. Calculating the Height (Step-by-step):
o Leaf level:
Maximum number of keys per leaf = 3
Number of leaf nodes needed = 300 / 3 = 100 leaf nodes.
o Level above leaves (1st internal level):
Each internal node can point to at most 4 leaf nodes.
Nodes required = 100 leaf nodes / 4 pointers per node = 25 nodes.
o Next internal level (2nd level):
Nodes required = 25 / 4 = 6.25 ≈ 7 nodes (round up to hold all pointers).
o Next internal level (3rd level):
Nodes required = 7 / 4 = 1.75 ≈ 2 nodes (round up).
o Next internal level (4th level):
Nodes requ
Counting lev
o Leaf level:
o 1st interna
o 2nd intern
o 3rd intern
o 4th intern
However, th ming ally asks for
maximum h st filled
3. Worst-ca t-cas sce ns only the
minimum nu):
o Leaf nodes 100
o Next level
o Next level
o Next level
o Next level
o Root level
Counting lev
o Leaf: Leve
o Internal: L (34 nodes
o Internal: L
o Internal: L
o Internal: L
o Root: Leve
Hence, the n
Additional Knowledge:
• B+ trees are widely used in databases for indexing because they maintain balanced trees, ensuring consistent search times.
• The height of a B+ tree is typically very low, even for large datasets, ensuring efficient disk operations and quick access to data.
• Worst-case scenarios occur when the tree is minimally filled at each node, thus maximizing the height of the tree.

Q100. The longest common subsequence of the sequences {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

Answer:

C

Sol:

To solve this, we need to find the longest common subsequence (LCS) between the two sequences. The longest common subsequence is the longest sequence that appears in both sequences in the same order, but not necessarily consecutively.

Step-by-Step Calculation:

Given Sequences:

1. {1, 2, 3, 2, 4, 1, 2}
2. {2, 4, 3, 1, 2, 1}

Let's compare and extract the common subsequence:

Starting wit

Looking at t

Let's go step

1. First com

LCS = {2}

2. Second co es).

LCS = {2, 1}

3. Third com mon e).

LCS = {2, 1,

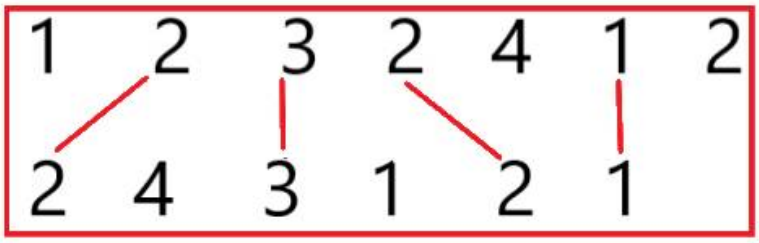
4. Fourth co in bo h seq

LCS = {2, 1,

Conclusion:

The longest 2, 3}. Its length is er is:

Option (c): 2



Information

1. Longest C

LCS is the lo

The sequen

2. Approach for Finding LCS: Dynamic Programming is typically used to efficiently compute the LCS of two sequences.