CUET-UG Computer Science Sample Paper - 6

Duration: 1 Hour **Maximum Marks: 250**

Instructions

- This paper contains a total of 50 Multiple Choice Questions.
- Each correct answer carries **+5 marks**.
- Each incorrect answer carries **-1 mark**.
- No negative marking for unattempted questions.

Q1. Consider the following SQL query: `SELECT ROUND(157.456, -2) FROM DUAL;`. What will be the output of this statement?

(A) 157.46

(B) 160

(C) 200

(D) 100

Q2. Given the string `'Artificial Intelligence'`, which SQL command will return `'cial'`?

(A) `SELECT SUBSTR('Artificial Intelligence', 6, 4);`

(B) `SELECT SUBSTR('Artificial Intelligence', 7, 4);`

(C) `SELECT MID('Artificial Intelligence', 6, 9);`

(D) `SELECT INSTR('Artificial Intelligence', 'cial');`

Q3. In SQL, which function is used to return the current system date and time?

(A) `DATE()`

(B) `GETDATE()`

(C) `NOW()`

(D) `CURRENT_TIME()`

Q4. What will be the output of `SELECT INSTR('CUET_EXAM_2026', '_');`?

(A) 4

(B) 5

(C) 10

(D) 0

Q5. A table 'Employee' has a column 'Salary'. Which query correctly displays the average salary rounded to 2 decimal places?

(A) `SELECT ROUND(AVG(Salary), 2) FROM Employee;`

(B) `SELECT AVG(ROUND(Salary, 2)) FROM Employee;`

(C) `SELECT ROUND(Salary, 2) FROM Employee;`

(D) `SELECT AVG(Salary) ROUND 2 FROM Employee;`

Q6. What value is returned by `SELECT DAYOFMONTH('2026-05-15');`?

(A) 5

(B) 2026

(C) 15

(D) *Friday*

Q7. Which mathematical function in SQL returns the remainder of a division?

(A) `DIV()`

(B) `REM()`

(C) `MOD()`

(D) `PERCENT()`

Q8. The SQL query `SELECT LCASE(LEFT('COMPUTER', 3));` results in:

(A) `com`

(B) `COM`

(C) `ter`

(D) TER

Q9. In Relational Algebra, which operation is used to select specific columns from a relation?

(A) Selection (σ)

(B) Projection (π)

(C) Cartesian Product ($\times$)

(D) Join ($\bowtie$)

Q10. A _____ is a minimal set of attributes that can uniquely identify a tuple in a relation.

(A) Primary Key

(B) Foreign Key

(C) Candidate Key

(D) Alternate Key

Q11. Given two relations R(A, B) and S(B, C), the Natural Join of R and S will have how many attributes?

(A) 4

(B) 3

(C) 2

(D) 5

Q12. Referential Integrity is maintained by which of the following constraints?

(A) Unique Key

(B) Check Constraint

(C) Foreign Key

(D) Primary Key

Q13. Which network topology requires a central controller or hub to connect all nodes?

(A) Mesh Topology

(B) Star Topology

(C) Bus Topology

(D) Ring Topology

Q14. A MAC address is a _____ bit physical address assigned to a Network Interface Card (NIC).

(A) 32

(B) 64

(C) 48

(D) 128

Q15. Which networking device operates at the Physical Layer of the OSI model and simply regenerates signals?

(A) Bridge

(B) Router

(C) Repeater

(D) Switch

Q16. Evaluate the following postfix expression: $10, 5, +, 3, *, 2, -$

(A) 43

(B) 45

(C) 13

(D) 33

Q17. Which data structure follows the LIFO (Last In First Out) principle?

(A) Queue

(B) Stack

(C) Linked List

(D) Tree

Q18. In a circular queue with size N, if `front` is 3 and `rear` is 2, how many elements are currently in the queue?

(A) $N - 1$

(B) 1

(C) 0

(D) $N - 2$

Q19. What is the result of converting the infix expression $(A + B) * C$ to prefix?

(A) $AB + C*$

(B) $* + ABC$

(C) $+ * ABC$

(D) $A + BC*$

Q20. In Python, what is the time complexity to append an element to the end of a list of size n?

(A) $O(1)$

(B) $O(n)$

(C) $O(\log n)$

(D) $O(n^2)$

Q21. Which of the following is NOT a valid operation on a Stack?

(A) Push

(B) Pop

(C) Peek

(D) Dequeue

Q22. What is the maximum number of elements that can be stored in a Stack implemented using a Python list if no size limit is defined?

(A) 2^{16}

(B) 1024

(C) Limited only by system memory

(D) 65536

Q23. If a stack contains elements $[10, 20, 30]$ where 30 is at the top, what will be the content of the stack after performing two `Pop()` operations and then one `Push(40)` operation?

(A) $[10, 20, 40]$

(B) $[10, 40]$

(C) $[40, 30, 20]$

(D) $[40, 10]$

Q24. The process of adding an element to a Queue is known as:

(A) Push

(B) Enqueue

(C) Dequeue

(D) Insert

Q25. A linear list of elements in which deletion can take place only from one end (front) and insertion can take place only at the other end (rear) is:

(A) Stack

(B) Queue

(C) Priority Queue

(D) Deque

Q26. Which Python method is used to remove and return the last element of a list?

(A) `delete()`

(B) `remove()`

(C) `pop()`

(D) `discard()`

Q27. What is the worst-case time complexity of Binary Search?

(A) $O(n)$

(B) $O(n^2)$

(C) $O(\log n)$

(D) $O(1)$

Q28. Which sorting algorithm repeatedly finds the minimum element from the unsorted part and puts it at the beginning?

(A) Bubble Sort

(B) Selection Sort

(C) Insertion Sort

(D) Merge Sort

Q29. How many passes are required to sort a list of 6 elements using Bubble Sort in the worst case?

(A) 6

(B) 5

(C) 36

(D) 12

Q30. The prerequisite for Binary Search is that the list must be:

(A) Unsorted

(B) Sorted

(C) Containing only integers

(D) Small in size

Q31. In Insertion Sort, the element is inserted into its correct position in a _____ subarray.

(A) Random

(B) Sorted

(C) Reversed

(D) Empty

Q32. In which case is Linear Search more efficient than Binary Search?

(A) When the array is very large and sorted.

(B) When the array is small and unsorted.

(C) When the element is at the end of a sorted list.

(D) It is never more efficient.

Q33. What is the total number of comparisons in the first pass of Bubble Sort for an array of size n?

(A) n

(B) n^2

(C) $n - 1$

(D) 1

Q34. Which algorithm has a best-case time complexity of $O(n)$ when the list is already sorted?

(A) Selection Sort

(B) Binary Search

(C) Bubble Sort (Optimized)

(D) All of the above

Q35. If the middle element is less than the target in Binary Search (Ascending Order), where should you search next?

(A) Left half

(B) Right half

(C) The search stops

(D) Check the first element

Q36. Which block in Python is executed whether an exception occurs or not?

(A) `catch`

(B) `except`

(C) `finally`

(D) `else`

Q37. What is the purpose of the `'a'` mode in the `open()` function in Python?

(A) Read only

(B) Overwrite the file

(C) Append to the end of the file

(D) Open for binary reading

(E)

Q38. Which method is used to read all lines from a file and return them as a list of strings?

(A) `read()`

(B) `readline()`

(C) `readlines()`

(D) `readall()`

Q39. Analyze the following Python code and determine the correct output:

```
try:
x = 20 / 0
except ZeroDivisionError:
print("Error", end=" ")
else:
print("Success", end=" ")
finally:
print("Finished")
```

(A) Error Success Finished

(B) Success Finished

(C) Error Finished

(D) Error

Q40. Which module is required for Binary File I/O (Serialization) in Python?

(A) `os`

(B) `sys`

(C) `pickle`

(D) `binary`

Q41. The `tell()` method in file handling returns:

(A) The name of the file

(B) The current position of the file pointer

(C) The total size of the file

(D) The number of lines in the file

Q42. Which exception is raised when a module is not found?

(A) `NameError`

(B) `ImportError`

(C) `ValueError`

(D) `KeyError`

Q43. The default mode for `open("data.txt")` is:

(A) `'w'`

(B) `'a'`

(C) `'r'`

(D) `'rb'`

Q44. Which protocol is used to transfer files between a client and a server on a computer network?

(A) HTTP

(B) SMTP

(C) FTP

(D) POP3

Q45. A _____ is a malicious program that attaches itself to another program and replicates when the program is executed.

(A) Worm

(B) Virus

(C) Trojan Horse

(D) Spyware

Q46. Which switching technique divides data into small units before transmission?

(A) Circuit Switching

(B) Message Switching

(C) Packet Switching

(D) Line Switching

Q47. What is the port number for HTTPS?

(A) 80

(B) 21

(C) 443

(D) 25

Q48. Which of the following is considered a cyber threat where a person is tricked into revealing sensitive information?

(A) Phishing

(B) Firewall

(C) Encryption

(D) Decryption

Q49. The `https://` in a URL stands for:

(A) Hyperlink Text Transfer Protocol Secure

(B) High Text Transfer Protocol Standard

(C) Hypertext Transfer Protocol Secure

(D) Hypertext Transmission Process System

Q50. Which layer of the OSI model provides end-to-end communication services for applications?

(A) Transport Layer

(B) Network Layer

(C) Application Layer

(D) Data Link Layer

Detailed Solutions

Q1.

Solution

Concept: This question tests the understanding of the SQL `ROUND()` function, which is used to round a numeric value to a specified number of decimal places. The syntax is `ROUND(number, decimals)`. A crucial aspect often tested in exams like CUET is the behavior of the function when the second argument (the decimal parameter) is a negative integer. While a positive integer rounds to the right of the decimal point, a negative integer rounds to the left (tens, hundreds, etc.).

Solution:

1. **Analyze the Input:** The given query is `SELECT ROUND(157.456, -2)`. Here, the number to be rounded is 157.456 and the rounding factor is -2.
2. **Identify the Position:** A value of -1 would round the number to the nearest 10. A value of -2 rounds the number to the nearest 100.
3. **Apply Rounding Logic:** We look at the digit at the hundreds position and the tens position. The number is 157.456. Since we are rounding to the nearest hundred (-2 position), we check the digit at the tens place, which is 5.
4. **Rule of Rounding:** In standard mathematical rounding used by SQL, if the digit to the right of the target position is 5 or greater, we round up. If it is less than 5, we round down.
5. **Calculation:** The tens digit is 5. Therefore, the value at the hundreds place (1) is incremented by 1, making it 2, and all digits to the right of the hundreds place become zero.
6. **Conclusion:** 157.456 rounded to the nearest hundred results in 200.

Final Answer: The output of the query will be 200.

Answer: (C)

Q2.

Solution

Concept: The `SUBSTR()` (or `SUBSTRING()`) function in SQL is used to extract a specific portion of a string. It typically takes three arguments: the source string, the starting position, and the number of characters to extract. It is a fundamental string manipulation function required for database management and data cleaning tasks.

Solution:

1. **Identify the Target:** The goal is to extract the substring `'cial'` from the main string `'Artificial Intelligence'`.

2. **Determine the Starting Position:** In SQL, string indexing usually starts at 1 (unlike Python, where it starts at 0). Let's count the characters:

A(1), r(2), t(3), i(4), f(5), i(6), c(7), i(8), a(9), l(10)...

Wait, let's recount carefully: A(1), r(2), t(3), i(4), f(5), i(6), **c(7)**, i(8), a(9), l(10).

The character 'c' in 'cial' is at the 7^{th} position.

3. **Determine the Length:** The substring `'cial'` consists of 4 characters ('c', 'i', 'a', 'l').

4. **Evaluate the Options:**

- Option (A) starts at position 6, which is 'i'. It would return 'icia'.
- Option (B) starts at position 7, which is 'c', and takes 4 characters. This results in 'cial'.
- Option (C) uses `MID()`, which is a synonym, but the length 9 is incorrect.
- Option (D) uses `INSTR()`, which returns the position of a substring, not the substring itself.

5. **Validation:** Since position 7 is 'c' and we need 4 characters, `SUBSTR(..., 7, 4)` is the correct syntax.

Final Answer: The correct SQL command is `SELECT SUBSTR('Artificial Intelligence', 7, 4);`.

Answer: (B)

Q3.

Solution

Concept: In Database Management Systems (DBMS), obtaining the current temporal data from the server is a frequent requirement. Different SQL dialects (MySQL, PostgreSQL, SQL Server, Oracle) have slightly different function names for this purpose, but CUET-UG primarily follows the standard MySQL/Informatic practices where `NOW()` is a primary function.

Solution:

1. **Function Roles:** We need to distinguish between functions that return only the date, only the time, or both.
2. **Analyze Option A:** `DATE()` is often used to extract the date part of a date/time expression, but it usually requires an argument or returns only the date in some systems.
3. **Analyze Option B:** `GETDATE()` is the standard function used in T-SQL (Microsoft SQL Server), but it is not the universal standard for all SQL environments taught in the core syllabus.
4. **Analyze Option C:** `NOW()` is the most common SQL function used to return the current date and time as a 'YYYY-MM-DD HH:MM:SS' format. It is widely used in MySQL, which is the primary reference for the curriculum.
5. **Analyze Option D:** `CURRENT_TIME()` usually returns only the time portion, not the date.
6. **Conclusion:** Because the question asks for both the current system date and time, `NOW()` is the most appropriate and standard answer within the context of the exam's expected environment.

Final Answer: The function used is `NOW()`.

Answer: (C)

Q4.

Solution

Concept: The `INSTR()` function (In-String) is used to find the location of the first occurrence of a substring within a larger string. If the substring is found, it returns the position of the first character of the first occurrence. If the substring is not found, it returns 0.

Solution:

1. **Define the Search:** We are looking for the underscore character `'_'` within the string `'CUET_EXAM_2026'`.

2. **Trace the String:** Let's map the characters to their index positions (starting from 1):

C → 1

U → 2

E → 3

T → 4

_ → 5

E → 6

X → 7

...and so on.

3. **Identify the First Match:** The `INSTR()` function stops searching as soon as it finds the very first instance of the target character/substring.

4. **Locate the Index:** The first underscore appears immediately after the 'T' in 'CUET'.

5. **Confirm the Position:** Counting from the start, 'C' is 1, 'U' is 2, 'E' is 3, 'T' is 4. The underscore is at the 5^{th} position.

6. **Result:** Even though there is another underscore at the 10^{th} position, the function returns only the index of the first occurrence.

Final Answer: The output will be 5.

Answer: (B)

Q5.

Solution

Concept: This question involves "Nested Functions" or "Function Composition" in SQL. We are combining an aggregate function `AVG()` (which calculates the mean of a numeric column) with a scalar function `ROUND()` (which formats the numeric result).

Solution:

1. **Requirement Analysis:** We need to calculate the average of the 'Salary' column and then format that result to show exactly 2 decimal places.
2. **Step 1: Calculate Average:** The `AVG(Salary)` function will compute the total sum of salaries divided by the count of records. This often results in a long floating-point number (e.g., 5432.12345).
3. **Step 2: Apply Rounding:** To limit this to 2 decimal places, we must wrap the result of the average inside the `ROUND()` function. The syntax becomes `ROUND(result, 2)`.
4. **Synthesize the Query:** Combining these, we get `ROUND(AVG(Salary), 2)`.
5. **Evaluate Alternatives:**

- `AVG(ROUND(Salary, 2))` would round every individual salary before averaging. While mathematically similar, it is less precise and not the standard way to format an aggregate result.
- Options C and D have syntax errors or miss the aggregate requirement entirely.

6. **Standard Practice:** In SQL, you always perform the calculation (aggregate) first and format the output (scalar) last for the best precision and readability.

Final Answer: The correct query is `SELECT ROUND(AVG(Salary), 2) FROM Employee;`.

Answer: (A)

Q6.

Solution

Concept: The `DAYOFMONTH()` function in SQL is a temporal function used to extract the numerical day of the month from a given date string or date object. It is part of a family of functions like `YEAR()`, `MONTH()`, and `DAYNAME()` that allow developers to decompose a full date into its individual components for reporting and data analysis.

Solution:

1. **Analyze the Date Format:** The input provided is `'2026-05-15'`. In SQL, the standard ISO date format is `YYYY-MM-DD`.

2. **Deconstruct the Date:**

- `YYYY` (Year): 2026
- `MM` (Month): 05 (May)
- `DD` (Day): 15

3. **Apply the Function:** The `DAYOFMONTH()` function specifically targets the `DD` part of the date. It ignores the year and the month entirely.

4. **Determine the Output:** The day value in the string is 15. Therefore, the function will return the integer 15.

5. **Comparison with other functions:**

- If the function was `MONTH()`, the output would be 5.
- If the function was `YEAR()`, the output would be 2026.
- If the function was `DAYNAME()`, the output would be `'Friday'`.

6. **Conclusion:** Since the specific function requested is `DAYOFMONTH()`, we only extract the day of the specific month regardless of the weekday or the year. This is useful for generating monthly reports or recurring billing cycles.

Final Answer: The value returned is 15.

Answer: (C)

Q7.

Solution

Concept: The Modulo operator is a fundamental mathematical tool in programming and database management. It is used to find the remainder of a division operation between two numbers. In SQL, this is typically represented by the `MOD()` function or the `%` operator, though `MOD(n, m)` is the standard functional approach.

Solution:

1. **Function Identification:** We are looking for a function that performs the operation $a \pmod{b}$. This means dividing a by b and returning what is left over.
2. **Analyze Option A:** `DIV()` is used for integer division. For example, 10 DIV 3 would result in 3, discarding the remainder.
3. **Analyze Option B:** `REM()` is used in some programming languages or specific SQL dialects (like Oracle), but it is not the primary standard function taught in general SQL curriculum for CUET.
4. **Analyze Option C:** `MOD()` is the standard SQL function. The syntax `MOD(N, M)` returns the remainder of N divided by M. For example, `SELECT MOD(10, 3)` returns 1.
5. **Analyze Option D:** `PERCENT()` is not a mathematical function for division in SQL; the `%` symbol is used for modulo in some systems, but the word `PERCENT` is not a function name.
6. **Conclusion:** The `MOD()` function is the correct mathematical function for calculating remainders. It is frequently used in logic to determine if a number is even or odd (e.g., `MOD(id, 2) = 0`).

Final Answer: The function is `MOD()`.

Answer: (C)

Q8.

Solution

Concept: This question involves the nesting of two string functions: `LEFT()` and `LCASE()` (or `LOWER()`). Function nesting requires evaluating the innermost function first and using its output as the input for the outer function. This demonstrates how data can be transformed in multiple stages within a single SQL query.

Solution:

1. **Identify the Inner Function:** The inner function is `LEFT('COMPUTER', 3)`.

2. **Evaluate the Inner Function:** The `LEFT()` function takes a string and a number n, and returns the first n characters from the left side of the string.

- String: 'COMPUTER'
- Number: 3
- Result: 'COM'

3. **Identify the Outer Function:** The outer function is `LCASE(...)`, where the input is now the result of the inner function ('COM').

4. **Evaluate the Outer Function:** The `LCASE()` (Lower Case) function converts all alphabetic characters in a string to lowercase.

- Input: 'COM'
- Transformation: 'C' → 'c', 'O' → 'o', 'M' → 'm'.
- Result: 'com'

5. **Final Assembly:** Combining both steps, the query takes the first three letters of 'COMPUTER' and then converts them to small letters.

6. **Comparison:** Option B ('COM') is incorrect because it misses the lowercase step. Option C and D ('ter' or 'TER') are incorrect because they represent the `RIGHT()` function logic.

Final Answer: The result is 'com'.

Answer: (A)

Q9.

Solution

Concept: Relational Algebra is a theoretical language used to model the operations performed on relational databases. It consists of several operators that take one or more relations as input and produce a new relation as output. Understanding the difference between Selection and Projection is fundamental to database theory.

Solution:

1. **Analyze the Requirement:** The question asks for the operation that selects **specific columns** (vertical subsets) from a table.
2. **Evaluate Selection (σ):** The Selection operator (σ) is used to filter rows (tuples) based on a specific condition. It creates a horizontal subset of the data. For example, "Select all students where Age > 18".
3. **Evaluate Projection (π):** The Projection operator (π) is used to select specific attributes or columns from a relation. It discards the columns not listed and removes duplicate rows from the resulting set. It creates a vertical subset of the data. For example, "Show only the Name and RollNo columns".
4. **Evaluate Cartesian Product ($\times$):** This operator combines every row of one table with every row of another. It does not filter columns or rows specifically based on a column-selection logic.
5. **Evaluate Join ($\bowtie$):** This combines related rows from two tables based on a common attribute.
6. **Conclusion:** Since we are extracting specific columns, the correct Relational Algebra operator is Projection, represented by the Greek letter Pi (π).

Final Answer: The operation is Projection (π).

Answer: (B)

Q10.

Solution

Concept: Database keys are attributes or sets of attributes that establish relationships between tables and ensure data integrity. A Candidate Key is a fundamental concept in normalization and table design, serving as a ”candidate” to become the Primary Key of the table.

Solution:

1. **Identify the Definition:** We are looking for a ”minimal set of attributes” that uniquely identifies a tuple. ”Minimal” means that no attribute can be removed from the set without losing the uniqueness property.
2. **Evaluate Primary Key:** A Primary Key is a *chosen* candidate key. While it does uniquely identify tuples, the definition of a ”set of attributes that *can* uniquely identify” generally points to the broader category of Candidate Keys first.
3. **Evaluate Candidate Key:** A Candidate Key is any attribute or minimal set of attributes that has the property of uniqueness and non-nullity. A table can have multiple Candidate Keys. From this pool, the Database Administrator selects one to be the Primary Key.
4. **Evaluate Foreign Key:** A Foreign Key is a column in one table that points to the Primary Key of another table. Its purpose is to maintain referential integrity, not unique identification within its own table.
5. **Evaluate Alternate Key:** Alternate Keys are Candidate Keys that were not chosen as the Primary Key.
6. **Conclusion:** The most accurate term for the definition provided is ”Candidate Key,” as it describes the inherent property of the attribute sets before one is specifically designated as the ”Primary” one.

Final Answer: The correct term is Candidate Key.

Answer: (C)

Q11.

Solution

Concept: The Natural Join ($\bowtie$) is a binary operator in relational algebra that creates a new relation by combining tuples from two relations based on common attributes. Unlike a standard Cartesian product or a Theta Join, the Natural Join is characterized by its "intelligence": it automatically identifies columns with the same name, performs an equality check, and most importantly, removes duplicate columns from the final result set.

Solution:

1. **Identify the Input Relations:** We are given Relation R with attributes $\{A, B\}$ and Relation S with attributes $\{B, C\}$.
2. **Identify Common Attributes:** The attribute 'B' exists in both R and S. This is the basis for the join.
3. **Process the Join:** The Natural Join will find all pairs of tuples in R and S such that the value of 'B' in R is equal to the value of 'B' in S.
4. **Apply Attribute Reduction:** In a standard join, we might see $(A, R.B, S.B, C)$. However, the definition of a Natural Join requires that common attributes appear only once. Therefore, $R.B$ and $S.B$ are merged into a single column 'B'.
5. **Formulate the Resulting Schema:** The set of attributes for the resulting relation is the union of the sets of attributes of the two relations. Symbolically: $\{A, B\} \cup \{B, C\} = \{A, B, C\}$.
6. **Count the Final Attributes:** By listing the distinct attributes (A, B, C), we find a total count of 3.
7. **General Rule Verification:** If R has n attributes and S has m attributes, and they share k common attributes, the Natural Join will always result in $n + m - k$ attributes. Here, $2 + 2 - 1 = 3$.

Final Answer: The Natural Join of R and S will have 3 attributes.

Answer: (B)

Q12.

Solution

Concept: Referential Integrity is a fundamental database constraint that ensures the consistency of data across multiple tables. It is based on the principle that if one table (the child/referencing table) refers to another table (the parent/referenced table), the reference must be valid. This prevents ”dangling references” or ”orphaned records” where a record points to a non-existent entity.

Solution:

1. **Define the Constraint Type:** Referential integrity is enforced through the use of Foreign Keys. A Foreign Key is a field (or collection of fields) in one table that uniquely identifies a row of another table.

2. **Mechanism of Action:** When you define a Foreign Key constraint, the database engine enforces rules: you cannot insert a value in the foreign key column if that value does not exist in the primary key of the referenced table.

3. **Evaluate Options:**

- **Unique Key:** Ensures all values in a column are unique but doesn't relate to other tables.
- **Check Constraint:** Ensures values meet a specific range or pattern (e.g., $age \geq 18$).
- **Primary Key:** Uniquely identifies a record within its *own* table.

4. **Integration:** The Foreign Key acts as the ”bridge.” It establishes a link between the data in two tables. This link is what allows the database to ”refer” from one table to the other, hence the term ”Referential.”

5. **Maintenance Operations:** Referential integrity also dictates what happens during deletions. If a parent record is deleted, the foreign key constraint can trigger a `CASCADE`, `SET NULL`, or `RESTRICT` action to maintain stability.

6. **Conclusion:** Therefore, the Foreign Key is the specific constraint dedicated to maintaining referential integrity.

Final Answer: Referential Integrity is maintained by the Foreign Key.

Answer: (C)

Q13.

Solution

Concept: Network Topology defines the layout of a computer network. Each topology varies in terms of physical design, cabling, and communication logic. One of the most common and robust topologies in modern Local Area Networks (LANs) is the Star Topology, which relies heavily on a centralized architecture for data distribution and management.

Solution:

1. **Analyze the Topology Requirement:** The question specifies a requirement for a ”central controller or hub” to facilitate connection between nodes.

2. **Review Star Topology:** In a Star network, every host (computer, printer, etc.) is connected to a central hub, switch, or router via a dedicated cable. The nodes do not communicate directly with each other; instead, data is sent to the central device, which then forwards it to the destination.

3. **Compare with Other Topologies:**

- **Mesh:** Every node connects to every other node; there is no center.
- **Bus:** All nodes share a single common cable (bus). No hub is required.
- **Ring:** Nodes are connected in a circular loop; data passes from one node to the next.

4. **Identify Key Characteristics:** The central device in a Star topology acts as a signal repeater or a bridge. This design simplifies troubleshooting: if one cable breaks, only one node goes offline. However, the central device is a single point of failure.

5. **Terminology Check:** Words like ”Hub,” ”Switch,” or ”Central Controller” are synonymous with the Star architecture in the context of CUET-UG Computer Science.

6. **Conclusion:** Because all nodes radiate from a central point, the Star topology is the only one that matches the description.

Final Answer: The Star Topology requires a central controller or hub.

Answer: (B)

Q14.

Solution

Concept: Network addressing happens at different layers of the OSI model. The Media Access Control (MAC) address is a hardware-level identifier, also known as a Physical Address. It is uniquely assigned to each Network Interface Card (NIC) by the manufacturer. Understanding the bit-length and structure of these addresses is vital for identifying data link layer operations.

Solution:

1. **Identify the Scale of MAC Addresses:** MAC addresses are typically represented as 12-digit hexadecimal numbers (e.g., $00 - 0C - 29 - 44 - AB - 01$).

2. **Perform Bit Conversion:**

- Each hexadecimal digit represents 4 bits in binary.
- Since there are 12 hexadecimal digits, the total number of bits is $12 \times 4 = 48$ bits.

3. **Address Structure:** The 48-bit address is split into two equal halves of 24 bits each.

- The first 24 bits are the *Organizationally Unique Identifier* (OUI), which identifies the manufacturer.
- The last 24 bits are the *Network Interface Controller* (NIC) specific identifier, assigned by the manufacturer.

4. **Comparison with IP Addressing:**

- IPv4 uses 32 bits (often confused with MAC).
- IPv6 uses 128 bits (used for modern global addressing).

5. **Application:** MAC addresses are used within the Data Link Layer (Layer 2) to deliver data packets (frames) between devices on the same local network segment. Unlike IP addresses, they are generally permanent.

6. **Conclusion:** The standard length for a MAC address across Ethernet and Wi-Fi hardware is strictly 48 bits.

Final Answer: A MAC address is a 48 bit physical address.

Answer: (C)

Q15.

Solution

Concept: Networking devices serve different purposes depending on the layer of the OSI model at which they operate. The Physical Layer (Layer 1) is concerned with the transmission and reception of raw bitstreams over a physical medium. Devices at this layer do not understand headers, IP addresses, or MAC addresses; they only deal with signals.

Solution:

1. **Examine Device Functionality:** The question describes a device that "operates at the Physical Layer" and "simply regenerates signals."

2. **Analyze the Repeater:** In networking, signal strength weakens over distance—a phenomenon called attenuation. A Repeater is used to combat this. It receives a signal before it becomes too weak or corrupted, cleans it of noise, amplifies it back to its original strength, and retransmits it.

3. **Layer Classification:**

- **Router:** Layer 3 (Network Layer). It processes IP packets and makes routing decisions.
- **Bridge/Switch:** Layer 2 (Data Link Layer). They process MAC addresses to forward frames.
- **Repeater/Hub:** Layer 1 (Physical Layer). They deal purely with electrical, optical, or radio signals.

4. **Distinguish from Hubs:** While a Hub is essentially a multi-port repeater, the most basic device described by the function of signal regeneration is the Repeater.

5. **Contextual Usage:** Repeaters are essential in long-distance fiber optic or ethernet runs where the physical limitations of the cable would otherwise prevent the signal from reaching the destination effectively.

6. **Conclusion:** The device that perfectly fits the description of a Layer 1 signal regenerator is the Repeater.

Final Answer: The networking device is a Repeater.

Answer: (C)

Q16.

Solution

Concept: Postfix notation, also known as Reverse Polish Notation (RPN), is a mathematical notation in which every operator follows all of its operands. It is widely used in computer science because it eliminates the need for parentheses and follows a strict order of operations using a stack. The evaluation algorithm involves scanning the expression from left to right: operands are pushed onto a stack, and when an operator is encountered, the required operands are popped, the operation is performed, and the result is pushed back.

Solution:

1. **Initial Expression:** $10, 5, +, 3, *, 2, -$
2. **Scan 10:** It is an operand. Push onto stack. [Stack: 10]
3. **Scan 5:** It is an operand. Push onto stack. [Stack: 10, 5]
4. **Scan +:** It is an operator. Pop two operands (5 and 10). Perform $10 + 5 = 15$. Push 15. [Stack: 15]
5. **Scan 3:** It is an operand. Push onto stack. [Stack: 15, 3]
6. **Scan *:** It is an operator. Pop two operands (3 and 15). Perform $15 \times 3 = 45$. Push 45. [Stack: 45]
7. **Scan 2:** It is an operand. Push onto stack. [Stack: 45, 2]
8. **Scan -:** It is an operator. Pop two operands (2 and 45). Perform $45 - 2 = 43$. Push 43. [Stack: 43]
9. **Final Result:** The expression is fully scanned, and the remaining value on the stack is 43.
10. **Verification:** In infix, this expression would be $((10 + 5) \times 3) - 2$, which equals $15 \times 3 - 2 = 45 - 2 = 43$.

Final Answer: The value of the postfix expression is 43.

Answer: (A)

Q17.

Solution

Concept: Data structures are organized based on how data is accessed and removed. Linear data structures like Stacks and Queues follow specific protocols. The LIFO (Last In First Out) principle dictates that the last element added to the collection must be the first one to be removed. This is analogous to a physical stack of plates; you add a plate to the top and must remove the top one first to avoid breaking the others.

Solution:

1. **Analyze LIFO:** Last In, First Out means the most recent entry is processed first.
2. **Evaluate Queue:** A Queue follows the FIFO (First In First Out) principle. Much like a line at a cinema, the person who arrived first is served first. Deletion happens at the front, and insertion happens at the rear.
3. **Evaluate Stack:** A Stack follows the LIFO principle. The operations are `Push` (to add an element to the top) and `Pop` (to remove an element from the top). Since both operations occur at the same end, the last item pushed is inevitably the first item popped.
4. **Evaluate Linked List:** A Linked List is a general-purpose linear data structure where elements (nodes) point to the next. It does not strictly enforce LIFO or FIFO unless it is specifically used to implement a Stack or a Queue.
5. **Evaluate Tree:** A Tree is a non-linear, hierarchical data structure. It does not follow LIFO/FIFO as a primary access rule.
6. **Conclusion:** The Stack is the definitive example of a LIFO data structure. It is used in function calls (the call stack), undo mechanisms in editors, and expression parsing.

Final Answer: The Stack follows the LIFO principle.

Answer: (B)

Q18.

Solution

Concept: A Circular Queue is an extended version of a regular queue where the last position is connected back to the first position to make a circle. This structure overcomes the limitation of "wasted space" in a linear queue implementation. Calculating the number of elements in a circular queue requires understanding how the `front` and `rear` pointers wrap around the fixed size of the array N.

Solution:

1. **Identify Variables:** We have a queue of size N, `front` = 3, and `rear` = 2.
2. **Analyze the Pointer Logic:** In a standard queue, `rear` is usually greater than or equal to `front`. When `rear` is less than `front`, it indicates that the queue has "wrapped around" the end of the array.
3. **Apply the Count Formula:** The general formula to find the number of elements in a circular queue is:

```
Count = (rear - front + N) % N
```

4. **Substitute the Values:**

```
Count = (2 - 3 + N) % N
Count = (N - 1) % N
```

5. **Step-by-Step Visualization:**

Positions filled: $3, 4, 5, \ldots, N-1$ (starting from front) AND $0, 1, 2$ (wrapping around to rear).
The only index *not* occupied is the one between the rear and the front (index 2 to 3 transition area).

6. **Logical Deduction:** If the queue were full, it would typically have $N-1$ elements (to distinguish between full and empty states in many implementations). Here, the pointers describe a state where almost all slots are filled except for a few gaps. Given the specific wrap-around (`rear` is exactly one index behind `front`), the queue is at its maximum capacity.
7. **Result:** In the standard circular queue model used in textbooks, if the `rear` is just behind the `front` after a wrap, the queue is considered "Full," holding $N-1$ elements.

Final Answer: There are $N-1$ elements in the queue.

Answer: (A)

Q19.

Solution

Concept: Prefix notation (Polish Notation) places the operator before its operands. Converting from Infix (human-readable) to Prefix requires following the order of precedence: Parentheses, Exponentiation, Multiplication/Division, and Addition/Subtraction. One common method for manual conversion is to fully parenthesize the expression, move the operators to the left of their respective parentheses, and then remove the parentheses.

Solution:

1. **Given Expression:** $(A + B) * C$
2. **Step 1 (Parentheses):** The expression $(A + B)$ must be processed first due to the explicit parentheses.
3. **Convert** $(A + B)$**:** Moving the '+' operator to the front of its operands results in $+AB$.
4. **Substitute back:** Now the expression looks like $[+AB] * C$.
5. **Step 2 (Multiplication):** Treat $[+AB]$ as a single operand and C as the second operand. The operator is '*'.
6. **Move Operator:** Place the '*' before both operands: $*[+AB][C]$.
7. **Final String:** Removing the brackets results in $* + ABC$.
8. **Alternative Method (Reverse-Post-Reverse):**

- Reverse infix: $C * (B + A)$
- Convert to postfix: $CBA + *$
- Reverse result: $* + ABC$.

9. **Validation:** Let's check the options. Option (A) is postfix. Option (C) is a different logic. Option (B) matches our derivation exactly.

Final Answer: The prefix expression is $* + ABC$.

Answer: (B)

Q20.

Solution

Concept: Time complexity (Big O notation) describes the computational effort required by an algorithm as the input size n increases. In Python, a 'list' is implemented as a dynamic array. This means it allocates a contiguous block of memory. Understanding the cost of various operations (like appending, inserting, or deleting) is crucial for writing efficient code and is a frequent topic in CUET-UG.

Solution:

1. **Understand the Operation:** The `append()` method adds an element to the end of a list.
2. **Analyze Dynamic Array Logic:** Python lists are "over-allocated." This means when you create a list, Python allocates more memory than is currently needed.
3. **Case 1 (Common):** If there is empty space at the end of the allocated memory block, adding the element is as simple as placing the value in the next slot and incrementing the size counter. This takes a constant amount of time, $O(1)$.
4. **Case 2 (Rare):** If the allocated space is completely full, Python must allocate a new, larger block of memory (usually double the size), copy all existing n elements to the new block, and then add the new element. This specific instance takes $O(n)$ time.
5. **Amortized Analysis:** Since Case 2 happens very infrequently (only when the list grows significantly), the "average" or "amortized" time complexity for an append operation is considered $O(1)$.
6. **Comparison:** Inserting an element at the beginning of a list (`list.insert(0, x)`) is $O(n)$ because every existing element must be shifted one position to the right. Appending at the end avoids this shifting.

Final Answer: The time complexity to append is $O(1)$.

Answer: (A)

Q21.

Solution

Concept: A Stack is an abstract data type that serves as a collection of elements with two principal operations: Push and Pop. It operates on the Last-In, First-Out (LIFO) principle. To master Stack theory, one must distinguish its native operations from those belonging to other data structures like Queues or Lists.

Solution:

1. **Analyze Push:** This is the standard operation to add an element to the top of the stack. It is a valid Stack operation.
2. **Analyze Pop:** This is the standard operation to remove the topmost element from the stack. It is a valid Stack operation.
3. **Analyze Peek:** Also known as 'Top', this operation allows a user to see the value of the topmost element without removing it from the stack. It is a valid Stack operation.
4. **Analyze Dequeue:** The term "Dequeue" specifically refers to the operation of removing an element from the **front** of a Queue. Since a Stack only allows access from one end (the top), and a Queue allows access from two ends (front and rear), the terminology is not interchangeable.
5. **Conclusion:** While you can technically write a function named `dequeue` in a Stack class, in the context of standard computer science nomenclature and competitive exams like CUET-UG, `Dequeue` is strictly a Queue operation.
6. **Comparison Table:** - Stack: Push, Pop, Peek, isEmpty. - Queue: Enqueue, Dequeue, Front, Rear.

Final Answer: Dequeue is NOT a valid operation on a Stack.

Answer: (D)

Q22.

Solution

Concept: In many programming languages (like C or Java), arrays and stacks have a fixed size defined at the time of creation (Static Stacks). However, Python lists are dynamic arrays. This means they can grow and shrink in size during runtime, leading to different constraints for Stack implementations.

Solution:

1. **Python Memory Management:** Python does not require you to declare the size of a list beforehand. When the current capacity of a list is reached, Python automatically allocates a larger block of memory.
2. **Theoretical Limit:** Mathematically, there is no fixed constant like 1024 or 65536 that limits a Python list. The size is only limited by the available RAM (Random Access Memory) of the computer system executing the code.
3. **Practical Constraint:** If a program keeps pushing elements into a stack without popping, it will eventually consume all available system memory, leading to a `MemoryError`.
4. **Evaluating Options:** - (A) and (D) are powers of 2 related to older 16-bit architectures, which do not apply to modern Python. - (B) is an arbitrary small number. - (C) correctly identifies that the limit is hardware-dependent.
5. **Conclusion:** A Stack implemented via a Python list is essentially "unbounded" from a software perspective, constrained only by the physical resources of the machine.

Final Answer: The maximum number of elements is limited only by system memory.

Answer: (C)

Q23.

Solution

Concept: This question tests the ability to track the state of a data structure through a sequence of operations. It requires a firm grasp of the LIFO (Last-In, First-Out) principle, where the most recently added item is the first to be removed.

Solution:

1. **Initial State:** The stack is $[10, 20, 30]$. The top of the stack is the last element, which is 30.
2. **Operation 1 (First Pop):** The `Pop()` operation removes the element at the top. - Element removed: 30. - Current Stack: $[10, 20]$.
3. **Operation 2 (Second Pop):** Another `Pop()` is performed. The current top is now 20. - Element removed: 20. - Current Stack: $[10]$.
4. **Operation 3 (Push 40):** The `Push(40)` operation adds the value 40 to the top of the existing stack. - Element added: 40. - Final Stack: $[10, 40]$.
5. **Logical Check:** Always ensure you are removing from the same side you are adding to. If the stack is represented as a list where the end of the list is the top, then $[10, 40]$ correctly shows 10 as the base and 40 as the new top.
6. **Conclusion:** After the two removals and one addition, only the original bottom element and the newly added element remain.

Final Answer: The stack will be $[10, 40]$.

Answer: (B)

Q24.

Solution

Concept: Terminology is a significant part of Computer Science exams. Different data structures use specific verbs to describe the insertion and deletion of data. For a Queue, which is a First-In, First-Out (FIFO) structure, the terminology is distinct from that of a Stack.

Solution:

1. **Defining the Operation:** The act of adding an item to a data structure is a generic ”insertion,” but in the context of a Queue, it has a formal name.
2. **Evaluate Push:** This term is exclusively used for Stacks. It refers to adding an item to the ”Top.”
3. **Evaluate Enqueue:** This is the correct technical term for adding an element to the ”Rear” of a Queue. It signifies that the element is joining the end of the line.
4. **Evaluate Dequeue:** This is the term for removing an element from the ”Front” of a Queue.
5. **Evaluate Insert:** While ”Insert” is a generic term used for lists (where you can add at any index), ”Enqueue” is the specific abstract data type (ADT) term for Queues.
6. **Conclusion:** When working with Queues in an academic or professional setting, adding an element is always referred to as ”Enqueueing” the data.

Final Answer: The process is known as Enqueue.

Answer: (B)

Q25.

Solution

Concept: A Queue is a linear data structure that models real-world lines (queues). It follows the FIFO (First-In, First-Out) principle. This means the first element added to the queue will be the first one to be removed. This requires two pointers: one to track the front (for deletions) and one to track the rear (for insertions).

Solution:

1. **Analyze the Movement:** In a standard Queue, data enters from one side and leaves from the opposite side.
2. **Front End:** This is where the ”oldest” data resides. All deletions (Dequeue operations) occur here.
3. **Rear End:** This is where ”new” data arrives. All insertions (Enqueue operations) occur here.
4. **Evaluate Options:** - **Stack:** In a stack, both insertion and deletion happen at the same end (the Top). This is LIFO. - **Queue:** This matches the description perfectly: deletion at the front, insertion at the rear. This is FIFO. - **Priority Queue:** In this structure, elements are removed based on priority, not necessarily the order they arrived at the front. - **Deque:** Short for ”Double-Ended Queue,” this allows insertion and deletion from **both** ends.
5. **Conclusion:** The description provided defines the fundamental behavior of a simple, linear Queue.

Final Answer: The data structure is a Queue.

Answer: (B)

Q26.

Solution

Concept: Python lists are highly versatile objects that come with several built-in methods for data manipulation. To manipulate a list as a Stack or a Queue, specific methods like `append()` and `pop()` are used. Understanding the difference between `remove()`, `pop()`, and `delete` is a common requirement in CUET-UG exams.

Solution:

1. **Analyze `delete()`:** There is no built-in method named `delete()` for Python lists. The keyword `del` exists, but it is a statement used to remove an item at a specific index or delete the entire list object, and it does not return the removed value.
2. **Analyze `remove()`:** This method searches for the first occurrence of a specific *value* and removes it. It does not take an index, and it returns `None`.
3. **Analyze `pop()`:** This is the standard method used to remove an element from a list. By default (if no index is provided), it removes and **returns** the last element of the list. This behavior is what allows Python lists to easily mimic a Stack's `Pop` operation.
4. **Analyze `discard()`:** This method is not available for lists; it is a method used with Sets to remove an element without raising an error if the element is not found.
5. **Functionality Check:** Because `pop()` both modifies the list and provides the value that was removed, it is the correct choice for the operation described.
6. **Conclusion:** For a list `L = [1, 2, 3]`, `L.pop()` will result in `L` becoming `[1, 2]` and will return the value `3`.

Final Answer: The method is `pop()`.

Answer: (C)

Q27.

Solution

Concept: Time complexity measures the efficiency of an algorithm by calculating the number of steps it takes relative to the input size n. Binary Search is an "Ordered Search" algorithm that significantly outperforms Linear Search on large datasets by employing a "divide and conquer" strategy.

Solution:

1. **Mechanism of Binary Search:** In each step of the algorithm, the search space is divided in half. If the target is not the middle element, the algorithm discards the half where the target cannot possibly be.

2. **Mathematical Derivation:** - After 1 step, the size is $n/2$. - After 2 steps, the size is $n/4$ $(n/2^2)$. - After k steps, the size is $n/2^k$. 3. **Worst-Case Scenario:** The worst case occurs when the search continues until only one element is left, or the element is not found at all. This happens when $n/2^k = 1$.

4. **Solving for k:** Taking the logarithm on both sides: $n = 2^k \implies \log_2(n) = k$.

5. **Comparison:** - **Linear Search:** $O(n)$ because it might check every single element. - **Binary Search:** $O(\log n)$ because it narrows the search space exponentially. 6. **Conclusion:** Because the number of comparisons grows very slowly as n increases, the time complexity is logarithmic.

Final Answer: The worst-case complexity is $O(\log n)$.

Answer: (C)

Q28.

Solution

Concept: Selection Sort is a simple comparison-based sorting algorithm. Its logic is based on dividing the list into two parts: a sorted sublist at the beginning and an unsorted sublist for the remainder. The algorithm essentially ”selects” the smallest (or largest) element from the unsorted portion in every pass.

Solution:

1. **Step-by-Step Logic:** - The algorithm starts at the first index. - It scans the entire remaining list to find the minimum value. - It swaps this minimum value with the value at the current starting index. - It then moves to the next index and repeats the process.
2. **Evaluate Bubble Sort:** This algorithm works by repeatedly swapping *adjacent* elements if they are in the wrong order. The ”heaviest” elements ”bubble” to the end.
3. **Evaluate Selection Sort:** This matches the question’s description. It explicitly looks for the minimum element in the unsorted part and places it at its correct position.
4. **Evaluate Insertion Sort:** This algorithm takes one element at a time and ”inserts” it into its correct position within a growing sorted prefix of the list.
5. **Key Identifier:** The phrase ”repeatedly finds the minimum element” is the defining characteristic of Selection Sort. This algorithm is known for making a minimum number of swaps ($O(n)$) even though it performs $O(n^2)$ comparisons.

Final Answer: The sorting algorithm is Selection Sort.

Answer: (B)

Q29.

Solution

Concept: Bubble Sort is a simple sorting algorithm that works by repeatedly stepping through the list, comparing adjacent elements, and swapping them if they are in the wrong order. The number of ”passes” is a key factor in determining the algorithm’s performance and is a frequent calculation in CS theory.

Solution:

1. **Define a ”Pass”:** A pass consists of a complete traversal through the unsorted portion of the list. After the first pass, the largest element is guaranteed to be at the final position.
2. **Worst-Case Requirement:** In the worst-case scenario (e.g., the list is in reverse order), how many times must we traverse to ensure every single element is correctly placed?
3. **Calculate for $n = 6$:** - Pass 1: Moves the largest to the 6th position. - Pass 2: Moves the 2nd largest to the 5th position. - Pass 3: Moves the 3rd largest to the 4th position. - Pass 4: Moves the 4th largest to the 3rd position. - Pass 5: Moves the 5th largest to the 2nd position.
4. **Conclusion:** Once 5 elements are in their correct positions, the 6th element is automatically in its correct position (the 1st place). Therefore, for a list of size n, the number of passes required is $n - 1$.
5. **Final Calculation:** For 6 elements, we need $6 - 1 = 5$ passes.
6. **Note:** While the total number of *comparisons* is $(n \times (n-1))/2$, the question specifically asks for the number of *passes*.

Final Answer: 5 passes are required.

Answer: (B)

Q30.

Solution

Concept: Binary Search is highly efficient, but it relies on a very strict mathematical assumption about the data structure. Unlike Linear Search, which can operate on any collection of data, Binary Search requires the data to have a predictable order to determine which half of the dataset to discard.

Solution:

1. **Examine the Logic:** Binary search works by comparing the target value to the middle element of the array.
2. **The Decision Rule:** - If Target < Middle: Move to the left half. - If Target > Middle: Move to the right half.
3. **The Prerequisite:** For the "Decision Rule" to work, the algorithm must be certain that all elements to the left of the middle are smaller, and all elements to the right are larger. This is only possible if the list is **Sorted**.
4. **What if it's Unsorted?** If the list is unsorted, there is no guarantee that the target isn't hiding in the "discarded" half. In such cases, Binary Search would fail to find existing elements.
5. **Evaluate Options:** - (A) Unsorted: Only Linear Search works here. - (B) Sorted: Mandatory for Binary Search. - (C) Integers: Not necessary; it works on strings or any comparable data types. - (D) Small size: While it works on small lists, its real value is in large lists.
6. **Conclusion:** The absolute prerequisite is that the data must be sorted in either ascending or descending order.

Final Answer: The list must be Sorted.

Answer: (B)

Q31.

Solution

Concept: Insertion Sort is an intuitive sorting algorithm that works similarly to the way you might sort playing cards in your hands. It splits the list into a ”sorted” part and an ”unsorted” part. Elements from the unsorted part are picked and placed into the correct position in the sorted part. This question focuses on identifying the nature of the sub-array where the insertion occurs.

Solution:

1. **Initial State:** At the start, the first element (index 0) is considered a sorted sub-array of size 1. The rest of the list is the unsorted part.
2. **The Insertion Process:** The algorithm takes the first element from the unsorted part (the ”key”) and compares it with elements in the sub-array to its left.
3. **Maintaining Order:** As the key is moved to the left, it shifts elements that are larger than it to the right. This shifting continues until the key finds its correct spot.
4. **Analyzing the Target Area:** The area to the left of the current key is, by definition, the part of the list that has already been processed and organized. Therefore, the key is always being inserted into a **Sorted** subarray.
5. **Growth:** With each iteration, the sorted subarray grows by one element, while the unsorted portion shrinks.
6. **Conclusion:** The efficiency of the algorithm depends on how ”unsorted” the original list is. If the subarray was not sorted, the algorithm would be unable to find the correct insertion point using simple comparisons.

Final Answer: The element is inserted into its correct position in a Sorted subarray.

Answer: (B)

Q32.

Solution

Concept: Efficiency in searching is often measured by the number of comparisons. While Binary Search is mathematically superior for large datasets ($O(\log n)$ vs $O(n)$), there are specific practical scenarios where Linear Search is preferred or naturally more efficient.

Solution:

1. **Prerequisites Analysis:** Binary Search *requires* a sorted list. Linear Search does not.
2. **Overhead Costs:** Binary Search requires calculating a midpoint, potentially sorting the list first (which is $O(n \log n)$), and specific memory access. For a very small list (e.g., 3 or 4 elements), the "overhead" of these operations might outweigh the simple sequential check of Linear Search.
3. **Unsorted Data:** If an array is unsorted, Binary Search cannot be used at all. In this context, Linear Search is not just "more efficient"; it is the only viable option between the two.
4. **Evaluating Option B:** When the array is small and unsorted, sorting it just to perform a Binary Search would be incredibly inefficient. Linear Search would find the element in a few steps without any preparation.
5. **Best Case Comparison:** Even in a sorted list, if the element is at the very first index, Linear Search finds it in 1 comparison, matching the best-case speed of more complex algorithms.
6. **Conclusion:** For small, unordered datasets, the simplicity and zero-setup nature of Linear Search make it the more efficient choice.

Final Answer: Linear Search is more efficient when the array is small and unsorted.

Answer: (B)

Q33.

Solution

Concept: Bubble Sort operates through a series of passes. In each pass, adjacent elements are compared and swapped if they are out of order. Understanding the exact number of comparisons in a specific pass is a fundamental aspect of algorithmic analysis in the CUET-UG Computer Science syllabus.

Solution:

1. **Analyze the First Pass:** In the first pass of Bubble Sort, the algorithm starts at index 0 and compares it with index 1, then index 1 with 2, and so on, until it reaches the end of the array.
2. **The Counting Logic:** - Comparison 1: Index 0 and 1 - Comparison 2: Index 1 and 2 - ... - Last Comparison: Index $(n-2)$ and $(n-1)$.
3. **Determine the Total:** If we have n elements, the number of adjacent pairs is exactly $n-1$.
4. **Effect of the First Pass:** By the time the first pass is complete, $(n-1)$ comparisons have been made, and the largest element has "bubbled up" to the very last position in the array $(n-1)$.
5. **Subsequent Passes:** In the second pass, we only need to make $n-2$ comparisons, because the last element is already in its correct spot.
6. **Conclusion:** For an array of size n, the very first traversal involves looking at every pair once, totaling $n-1$ operations.

Final Answer: The total number of comparisons in the first pass is $n-1$.

Answer: (C)

Q34.

Solution

Concept: Best-case time complexity refers to the minimum number of steps an algorithm takes when the input is already in its most favorable state (usually already sorted). Different sorting and searching algorithms respond differently to ”perfect” data.

Solution:

1. **Evaluate Selection Sort:** Selection Sort always scans the entire remaining unsorted list to find the minimum, regardless of whether the list is sorted. Its complexity is always $O(n^2)$.
2. **Evaluate Binary Search:** While efficient, Binary Search still takes $O(\log n)$ to confirm an element's position, or $O(1)$ only if the middle element happens to be the target. However, ”best case $O(n)$” is not a standard descriptor for Binary Search.
3. **Evaluate Optimized Bubble Sort:** Standard Bubble Sort is $O(n^2)$. However, if we add a ”flag” to check if any swaps occurred during a pass, the algorithm can stop early. If the list is already sorted, the first pass will complete with zero swaps, and the algorithm will terminate.
4. **Calculate Complexity:** In this optimized scenario, the algorithm makes $n-1$ comparisons in the first pass and then stops. This results in a best-case time complexity of $O(n)$.
5. **Contrast with others:** Insertion Sort also has an $O(n)$ best case, but among the given options, Optimized Bubble Sort is the prominent example taught in this context.
6. **Conclusion:** Through the use of a swap-detecting flag, Bubble Sort becomes highly efficient for already sorted data.

Final Answer: Optimized Bubble Sort has a best-case complexity of $O(n)$.

Answer: (C)

Q35.

Solution

Concept: Binary Search relies on a "Divide and Conquer" approach. The search space is halved in every step based on a comparison between the target and the midpoint. The direction in which the search continues depends on whether the list is in ascending or descending order and how the target relates to the middle value.

Solution:

1. **Initial Setup:** Assume we are searching for a target value T in a list sorted in **Ascending Order** (smallest to largest).
2. **Midpoint Comparison:** The algorithm identifies the middle element M.
3. **Analyze the Condition:** The question states: "The middle element is less than the target" ($M < T$).
4. **Determine the Logic:** - Since the list is sorted ascendingly, all elements to the *left* of M are even smaller than M. - If M is already smaller than T, then everything to the left of M is also guaranteed to be smaller than T. - Therefore, the target T *must* be located in the portion of the list to the **right** of the middle element.
5. **Action Taken:** The algorithm updates its "low" pointer to $mid + 1$ and continues the search in the right half.
6. **Conclusion:** In an ascending list, if your current middle point is too low, you must move toward the higher values, which are found in the right half of the array.

Final Answer: You should search the Right half next.

Answer: (B)

Q36.

Solution

Concept: Exception handling in Python is managed through four main blocks: `try`, `except`, `else`, and `finally`. While `except` and `else` are conditional, the `finally` block is designed to provide a "guaranteed execution" path, which is critical for cleaning up resources such as closing database connections or open files.

Solution:

1. **Analyze the Try block:** This block contains the code that might raise an exception.
2. **Analyze the Except block:** This block only executes if an error occurs within the `try` block. It is used for error recovery.
3. **Analyze the Else block:** This block only executes if the `try` block finishes successfully without raising any exceptions.
4. **Analyze the Finally block:** This block is unique because it executes **no matter what**. Whether an exception was raised, caught, or even if the program encountered a `return` statement, the code inside `finally` will run.
5. **Comparison with other languages:** In Java or C#, this behaves exactly the same way. It is often called the "clean-up" block.
6. **Conclusion:** Because the question asks for the block that executes regardless of whether an exception occurs or not, `finally` is the only correct answer.

Final Answer: The `finally` block is executed whether an exception occurs or not.

Answer: (C)

Q37.

Solution

Concept: File handling in Python involves opening files in specific "modes" that determine what operations (read, write, append) can be performed. The `'a'` mode stands for "Append" and is a vital tool for logging or adding data to existing files without destroying previous content.

Solution:

1. **Analyze 'r' mode:** Opens a file for reading. If the file doesn't exist, it raises an error.
2. **Analyze 'w' mode:** Opens a file for writing. If the file exists, it **truncates** (deletes) all existing content first. If it doesn't exist, it creates a new one.
3. **Analyze 'a' mode:** Opens a file for appending. The file pointer is placed at the **end of the file**. Any data written to the file using this mode will be added after the existing data.
4. **Handling Non-existent Files:** Like 'w' mode, if the file specified in 'a' mode does not exist, Python will create a new empty file.
5. **Comparison with 'w':** The primary difference is that 'a' preserves existing data, while 'w' overwrites it.
6. **Conclusion:** The purpose of the `'a'` mode is to append data to the end of the file.

Final Answer: The `'a'` mode is used to append to the end of the file.

Answer: (C)

Q38.

Solution

Concept: Python provides multiple methods to retrieve data from a text file. Choosing the right method depends on how you intend to process the data. The `readlines()` method is specifically designed for cases where you need to manipulate individual lines as distinct elements in a collection.

Solution:

1. **Analyze `read()`:** This method reads the entire content of the file and returns it as a **single large string**. This includes all newline characters ($\backslash n$).
2. **Analyze `readline()`:** This method reads exactly **one line** from the file (up to the next $\backslash n$ character) and returns it as a string.
3. **Analyze `readlines()`:** This method reads **all lines** from the file from the current pointer position to the end and returns them as a **list of strings**. Each string in the list represents one line from the file.
4. **Example:** If a file has three lines, `readlines()` returns `["Line 1\n", "Line 2\n", "Line 3\n"]`.
5. **Use Case:** This is very useful for iterating through a file using a `for` loop or accessing specific lines via list indexing (e.g., `lines[0]`).
6. **Conclusion:** Because the question specifies returning "all lines" as a "list of strings," `readlines()` is the correct method.

Final Answer: The method is `readlines()`.

Answer: (C)

Q39.

Solution

Concept: This question tests the logical flow of the `try-except-else-finally` structure. Understanding which blocks are mutually exclusive and which are mandatory is essential for predicting the output of Python scripts.

Solution:

1. **Trace the Try Block:** The code executes `x = 20 / 0`. Division by zero is a prohibited mathematical operation in Python.
2. **Exception Identification:** This operation immediately raises a `ZeroDivisionError`. The rest of the code inside the `try` block is skipped.
3. **Match the Except Block:** Python looks for a matching `except` handler. Since we have `except ZeroDivisionError:`, this block is entered.
4. **Except Execution:** The code prints `"Error"` followed by a space (due to `end=" "`).
5. **Evaluate the Else Block:** The `else` block **only** runs if the `try` block completed successfully. Since an error occurred, the `else` block is entirely ignored.
6. **Evaluate the Finally Block:** The `finally` block **always** runs. It prints `"Finished"`.
7. **Final Output String:** Combining the results from the executed blocks, we get `"Error Finished"`.

Final Answer: The output is ”Error Finished”.

Answer: (C)

Q40.

Solution

Concept: Serialization is the process of converting complex data structures (like lists, dictionaries, or class objects) into a byte stream so they can be saved to a binary file or transmitted over a network. In Python, the standard module for this ”pickling” and ”unpickling” process is the `pickle` module.

Solution:

1. **Analyze Text vs Binary:** Text files store characters. Binary files store raw bytes. You cannot simply write a Python list to a text file without converting it to a string first.
2. **The Role of Pickle:** The `pickle` module allows you to save the entire object structure. When you ”unpickle” it later, you get back the exact same Python object.
3. **Module Options:**

- **os:** Used for interacting with the operating system (e.g., deleting files, changing directories).
- **sys:** Used for system-specific parameters and functions (e.g., command line arguments).
- **pickle:** The dedicated module for binary serialization in Python.
- **binary:** Not a standard Python module name for file I/O.

4. **Key Functions:** The two primary functions are `pickle.dump()` (to write to a file) and `pickle.load()` (to read from a file).
5. **Conclusion:** For handling binary file I/O and object serialization, the `pickle` module is mandatory.

Final Answer: The `pickle` module is required.

Answer: (C)

Q41.

Solution

Concept: In file handling, the file pointer (or cursor) determines the location from which the next read or write operation will begin. Managing this pointer is essential for random access within a file. Python provides two primary methods for this: `seek()` to move the pointer and `tell()` to locate it.

Solution:

1. **Analyze File Operations:** When a file is opened, a pointer is placed at a specific byte offset (usually 0 for read/write or at the end for append).

2. **Function of `tell()`:** As you read or write data, the pointer moves forward by the number of bytes processed. The `tell()` method returns an integer representing the current byte position of the pointer from the beginning of the file.

3. **Evaluate Options:**

- **Option A:** The filename is usually accessed via the `.name` attribute of the file object.
- **Option B:** This is the exact definition of `tell()`. If you have read 10 characters, `tell()` will return 10.
- **Option C:** The total size is often found using `os.path.getsize()` or by seeking to the end and then calling `tell()`.
- **Option D:** To count lines, one would typically use `len(f.readlines())`.

4. **Practical Example:** If you open a file and read the first 5 bytes, calling `f.tell()` will output 5, indicating you are ready to read from the 6th byte.

5. **Conclusion:** The `tell()` method is the standard way to retrieve the current cursor position in a binary or text stream.

Final Answer: The `tell()` method returns the current position of the file pointer.

Answer: (B)

Q42.

Solution

Concept: Python's modularity allows developers to use code from external files. When an `import` statement is executed, Python searches through a specific set of directories (defined in `sys.path`). If it fails to find the specified module, it raises a specific built-in exception.

Solution:

1. **Analyze Exception Types:** Python has a hierarchy of exceptions to help developers debug specific errors accurately.
2. **Evaluate `NameError`:** This occurs when a local or global name is not found (e.g., calling a variable that hasn't been defined).
3. **Evaluate `ImportError`:** This is the base exception raised when an `import` statement fails to load a module. A more specific subclass is `ModuleNotFoundError`. In most CUET-level contexts, `ImportError` is the primary answer for failed imports.
4. **Evaluate `ValueError`:** This occurs when a function receives an argument of the correct type but an inappropriate value (e.g., `int("abc")`).
5. **Evaluate `KeyError`:** This is raised when a dictionary key is not found.
6. **Conclusion:** Because the error specifically pertains to the inability to locate a module or a name within a module during the import process, `ImportError` (or its specific subclass) is the correct exception.

Final Answer: The exception raised is `ImportError`.

Answer: (B)

Q43.

Solution

Concept: The `open()` function is the gateway to file manipulation in Python. It requires the file path and an optional mode. If the mode is omitted, Python assumes a set of default behaviors designed for the most common use case: safe reading of text data.

Solution:

1. **Identify Default Behavior:** Python's design philosophy prioritizes safety. Therefore, it defaults to "Reading" mode to prevent accidental deletion or modification of data.
2. **Identify Data Type:** The default data format is "Text" (`t`), not binary (`b`).
3. **Combine the Modes:** The default mode for `open()` is `'rt'`, which stands for "Read Text." However, the `'t'` is usually implicit, so the standard shorthand is simply `'r'`.
4. **Evaluate Options:**

- `'w'` (Write) is not the default because it would overwrite existing files immediately.
- `'a'` (Append) is not the default as most users intend to read data first.
- `'rb'` is for binary reading, used for images or pickled files, but not the general default.

5. **Conclusion:** When you write `f = open("data.txt")`, Python executes it as if you had written `f = open("data.txt", "r")`.

Final Answer: The default mode is `'r'`.

Answer: (C)

Q44.

Solution

Concept: Networking protocols are sets of rules that govern how data is exchanged across a network. Different protocols operate at the Application Layer of the OSI model to handle specific tasks like web browsing, email transmission, or file management.

Solution:

1. **Analyze Protocol Roles:**

- **HTTP (Hypertext Transfer Protocol):** Used for transferring web pages and related data from a server to a browser.
- **SMTP (Simple Mail Transfer Protocol):** Used specifically for sending emails from a client to a server or between servers.
- **FTP (File Transfer Protocol):** This protocol was explicitly designed for the robust transfer of files (uploading and downloading) between a client and a server.
- **POP3 (Post Office Protocol v3):** Used by email clients to retrieve emails from a mail server.

2. **Identify the Best Fit:** While HTTP can transfer files, FTP is the dedicated protocol that supports features like directory browsing, resuming interrupted transfers, and handling large binary data efficiently.

3. **Conclusion:** For the specific task of ”transferring files between a client and a server,” FTP is the correct technical answer.

Final Answer: The protocol is FTP.

Answer: (C)

Q45.

Solution

Concept: Cyber threats come in various forms, collectively known as malware. Understanding the distinction between viruses, worms, and Trojans is a core requirement for the Data Communication and Security segment of the CUET-UG syllabus.

Solution:

1. **Analyze the Virus:** A virus is a piece of code that is ”parasitic.” It cannot run on its own; it must attach itself to a host file (like an `.exe` or `.docx` file). When a user runs that file, the virus code executes and replicates by infecting other files on the system.
2. **Analyze the Worm:** A worm is a standalone program that replicates itself across networks without needing a host file or human intervention.
3. **Analyze the Trojan Horse:** This is malware that disguises itself as legitimate software. It does not necessarily ”replicate” by attaching to other programs; it relies on tricking the user into installing it.
4. **Identify the Match:** The question specifically mentions ”attaches itself to another program” and ”replicates when the program is executed.” This is the classic, textbook definition of a computer Virus.
5. **Conclusion:** The defining characteristic of a virus is its dependency on a host program for propagation.

Final Answer: The program is a Virus.

Answer: (B)

Q46.

Solution

Concept: Switching techniques determine how data travels from source to destination across a network. The evolution from circuit switching (used in old telephony) to modern packet switching is what made the Internet efficient and scalable.

Solution:

1. **Circuit Switching:** In this method, a dedicated physical path is established between two nodes for the duration of the communication. It does not divide data into units; the whole stream flows through the reserved "circuit."
2. **Message Switching:** The entire data message is treated as a single unit and transferred from node to node (store-and-forward). There is no division into smaller chunks.
3. **Packet Switching:** This is the foundation of modern networking (TCP/IP). Large data files are broken down into smaller, manageable units called **packets**. Each packet contains a portion of the data plus a header with destination information. Packets can take different paths to reach the same destination and are reassembled upon arrival.
4. **Advantages:** Packet switching is more efficient because it doesn't tie up a whole line and allows multiple users to share the same network resources simultaneously.
5. **Conclusion:** The technique that "divides data into small units" is Packet Switching.

Final Answer: The technique is Packet Switching.

Answer: (C)

Q47.

Solution

Concept: Port numbers are part of the transport layer addressing (TCP/UDP) that help direct network traffic to the correct application on a server. Well-known ports (0–1023) are reserved for standard protocols.

Solution:

1. **Identify the Protocols:**

- **Port 80:** Standard for HTTP (unencrypted web traffic).
- **Port 21:** Standard for FTP (File Transfer Protocol).
- **Port 25:** Standard for SMTP (Email transmission).
- **Port 443:** Standard for HTTPS (Hypertext Transfer Protocol Secure).

2. **The Importance of 443:** HTTPS uses SSL/TLS to encrypt data between the browser and the server. To distinguish this secure traffic from regular web traffic, a different port is used. Port 443 is the globally recognized port for this purpose.
3. **Conclusion:** When you see a URL starting with `https://`, your browser is communicating over port 443.

Final Answer: The port number for HTTPS is 443.

Answer: (C)

Q48.

Solution

Concept: Social engineering is a type of cyberattack that exploits human psychology rather than technical vulnerabilities. Phishing is the most prevalent form of social engineering used to steal credentials and financial information.

Solution:

1. **Analyze Phishing:** This attack typically involves fraudulent emails or websites that mimic legitimate entities (like banks or social media sites). The goal is to trick the user into typing in their username, password, or credit card details.

2. **Evaluate Other Options:**

- **Firewall:** A security system that monitors and controls incoming and outgoing network traffic. It is a protective measure, not a threat.

- **Encryption/Decryption:** Mathematical processes used to secure data. These are security tools.

3. **Identify the Threat:** The description "tricked into revealing sensitive information" perfectly describes the mechanism of Phishing. It relies on the "human element" being the weakest link in the security chain.

4. **Conclusion:** Phishing remains a primary concern in cyber security education because it bypasses technical defenses like firewalls by targeting the user directly.

Final Answer: The threat described is Phishing.

Answer: (A)

Q49.

Solution

Concept: A URL (Uniform Resource Locator) is the address used to access resources on the Internet. It consists of several parts: the protocol, the domain name, and the path. Understanding the acronyms within a URL is a basic literacy requirement in Computer Science.

Solution:

1. **Break down the acronym:** - **H**: Hypertext (refers to the structured text/links used on the web). - **T**: Transfer (describes the action of moving data). - **T**: Transmission is incorrect; it is **Transfer**. - **P**: Protocol (a set of rules). - **S**: Secure (indicates the use of SSL/TLS encryption).

2. **Evaluate the options:**

- Option A: "Hyperlink" is incorrect. - Option B: "High" is incorrect. - Option C: "Hypertext Transfer Protocol Secure" is the standard, globally accepted definition.

3. **Functionality:** HTTPS ensures that the data sent between the user's browser and the website cannot be read by eavesdroppers, protecting privacy and integrity.

Final Answer: HTTPS stands for Hypertext Transfer Protocol Secure.

Answer: (C)

Q50.

Solution

Concept: The OSI (Open Systems Interconnection) model is a conceptual framework that standardizes the functions of a telecommunication system into seven layers. The top layers deal with application issues, while the lower layers deal with data transport issues.

Solution:

1. **Analyze the Application Layer (Layer 7):** This is the layer closest to the end user. It provides the interface between the software application (like a browser or email client) and the network. It provides services like file transfers, email, and web browsing.
2. **Analyze the Transport Layer (Layer 4):** This layer is responsible for end-to-end communication, error recovery, and flow control. It ensures that data is delivered reliably (TCP) or quickly (UDP).
3. **Identify the Match:** The question asks for the layer that provides communication services "for applications." This points directly to the Application Layer.
4. **Comparison:** While the Transport Layer handles the "how" of the data movement, the Application Layer handles the "what" for the user's software.
5. **Conclusion:** Therefore, the Application Layer is the correct classification for providing network services to user applications.

Final Answer: The Application Layer provides these services.

Answer: (C)

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans	Q	Ans
1	C	2	B	3	C	4	B	5	A
6	C	7	C	8	A	9	B	10	C
11	B	12	C	13	B	14	C	15	C
16	A	17	B	18	A	19	B	20	A
21	D	22	C	23	B	24	B	25	B
26	C	27	C	28	B	29	B	30	B
31	B	32	B	33	C	34	C	35	B
36	C	37	C	38	C	39	C	40	C
41	B	42	B	43	C	44	C	45	B
46	C	47	C	48	A	49	C	50	C