CUET-UG Computer Science Sample Paper - 11

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. Predict the output of the following SQL statement: `SELECT TRUNCATE(275.864, 1) FROM DUAL;`

(A) 275.9

(B) 275.8

(C) 276

(D) 275.86

Q2. Which SQL function is used to remove leading and trailing spaces from a string?

(A) `CLEAN()`

(B) `STRIP()`

(C) `TRIM()`

(D) `REMOVE()`

Q3. What will be the result of `SELECT POW(4, 3);` in a standard SQL environment?

(A) 12

(B) 16

(C) 64

(D) 81

Q4. The function `MONTHNAME('2026-08-15')` will return which of the following values?

(A) `8`

(B) `August`

(C) `Aug`

(D) `Saturday`

Q5. Which of the following SQL functions can be used to find the total number of characters in a string 'DATABASE'?

(A) `COUNT()`

(B) `SIZE()`

(C) `LENGTH()`

(D) `WIDTH()`

Q6. What is the output of `SELECT MID('Global Warming', 8, 4);`?

(A) `Warm`

(B) `al W`

(C) `armi`

(D) `Warming`

Q7. If you want to display the current date only (without time) in SQL, which function would you use?

(A) `NOW()`

(B) `CURDATE()`

(C) `SYSTIME()`

(D) `TODAY()`

Q8. What does `SELECT ROUND(1234.5678, -1);` return?

(A) 1230

(B) 1234.6

(C) 1235

(D) 1240

Q9. Which operator in Relational Algebra is used to combine all rows from two relations while removing duplicates, provided both relations have the same schema?

(A) Intersection ($\cap$)

(B) Union ($\cup$)

(C) Set Difference ($-$)

(D) Cartesian Product ($\times$)

Q10. In a database, a Candidate Key that is not selected as the Primary Key is called a/an _____.

(A) Secondary Key

(B) Foreign Key

(C) Alternate Key

(D) Composite Key

Q11. The degree of a relation refers to the _____ in that relation.

(A) Number of Tuples

(B) Number of Attributes

(C) Number of Constraints

(D) Number of Keys

Q12. Which property ensures that a transaction is treated as a single, indivisible unit that either succeeds completely or fails completely?

(A) Consistency

(B) Isolation

(C) Atomicity

(D) Durability

Q13. In which type of network is every computer connected to a single central cable using connectors?

(A) Ring Topology

(B) Bus Topology

(C) Star Topology

(D) Mesh Topology

Q14. Which protocol is used to assign dynamic IP addresses to devices on a network?

(A) DNS

(B) HTTP

(C) DHCP

(D) SMTP

Q15. An IPv4 address is composed of how many bits?

(A) 16

(B) 32

(C) 48

(D) 128

Q16. Which operation in a Stack is used to check the top element without removing it?

(A) `Push`

(B) `Pop`

(C) `Peek`

(D) `Overflow`

Q17. What is the postfix form of the infix expression: $A + B * C$?

(A) $AB + C*$

(B) $ABC * +$

(C) $A + BC*$

(D) $+A * BC$

Q18. A _____ is a linear data structure where elements are added at one end and removed from the same end.

(A) Queue

(B) Linked List

(C) Stack

(D) Tree

Q19. In a Queue implemented using a list, if we perform `L.pop(0)`, which element is removed?

(A) The last element

(B) The first element

(C) The middle element

(D) No element is removed

Q20. What is the result of evaluating the postfix expression: $8, 4, /, 2, +$?

(A) 4

(B) 6

(C) 3

(D) 10

Q21. Which data structure is most suitable for implementing a ”Undo” feature in a text editor?

(A) Queue

(B) Stack

(C) Array

(D) Tree

Q22. The situation where a `Pop` operation is attempted on an empty stack is known as _____.

(A) Overflow

(B) Underflow

(C) Emptyflow

(D) Garbage Collection

Q23. How many pointers are primarily required to manage a linear Queue?

(A) 1

(B) 2

(C) 3

(D) 4

Q24. Which of the following is a non-linear data structure?

(A) Stack

(B) Queue

(C) Tree

(D) List

Q25. Convert the prefix expression $+ * ABC$ to infix:

(A) $A * B + C$

(B) $(A * B) + C$

(C) $A + (B * C)$

(D) $A * (B + C)$

Q26. In Python, which list method is functionally equivalent to an `Enqueue` operation?

(A) `insert()`

(B) `remove()`

(C) `append()`

(D) `extend()`

Q27. What is the maximum number of comparisons needed for Linear Search in a list of 100 elements?

(A) 1

(B) 7

(C) 50

(D) 100

Q28. Which sorting algorithm is based on the ”Divide and Conquer” strategy and uses a pivot element?

(A) Bubble Sort

(B) Selection Sort

(C) Quick Sort

(D) Insertion Sort

Q29. After the first pass of Bubble Sort on the list $[5, 1, 4, 2, 8]$, what will be the list?

(A) $[1, 4, 2, 5, 8]$

(B) $[1, 5, 4, 2, 8]$

(C) $[1, 2, 4, 5, 8]$

(D) $[5, 4, 2, 1, 8]$

Q30. The time complexity $O(n^2)$ is common for which of the following in the worst case?

(A) Binary Search

(B) Bubble Sort

(C) Linear Search

(D) Both (A) and (B)

Q31. In Selection Sort, if the list has n elements, how many total swaps are performed in the worst case?

(A) n^2

(B) $n(n-1)/2$

(C) $n-1$

(D) n

Q32. Which searching algorithm is faster for a sorted list of $1,000,000$ elements?

(A) Linear Search

(B) Binary Search

(C) Both are equal

(D) Selection Search

Q33. What is the 'best-case' scenario for Linear Search?

(A) Element is at the end

(B) Element is not in the list

(C) Element is at the first position

(D) Element is in the middle

Q34. Which sorting algorithm maintains a "sorted" and "unsorted" sub-array and picks the first element of the unsorted part to place it correctly?

(A) Selection Sort

(B) Insertion Sort

(C) Bubble Sort

(D) Quick Sort

Q35. If a list is already sorted, which sorting algorithm (without optimization) still takes $O(n^2)$ time?

(A) Selection Sort

(B) Insertion Sort

(C) Bubble Sort

(D) All of the above

Q36. What is the output of the following code? `x = [1, 2, 3]`

```
try:
print(x[5])
except IndexError:
print("Index Error", end=" ")
finally:
print("Done")
```

(A) Index Error

(B) Done

(C) Index Error Done

(D) No output

Q37. To open a file for both reading and writing in Python, which mode is used?

(A) `'rw'`

(B) `'r+'`

(C) `'w'`

(D) `'a'`

Q38. Which function is used to write a single string to a file?

(A) `write()`

(B) `writelines()`

(C) `put()`

(D) `output()`

Q39. Which keyword is used to manually raise an exception in Python?

(A) `stop`

(B) `error`

(C) `raise`

(D) `throw`

Q40. When using `pickle.load(file_object)`, the file must be opened in which mode?

(A) `'r'`

(B) `'w'`

(C) `'rb'`

(D) `'wb'`

Q41. What will `f.seek(0, 2)` do if `f` is a file object?

(A) Move pointer to the beginning

(B) Move pointer to the end of the file

(C) Move pointer to the second character

(D) Move pointer to the current position

Q42. Which of the following is a standard Python Exception raised when a dictionary key is not found?

(A) `ValueError`

(B) `KeyError`

(C) `LookupError`

(D) `NameError`

Q43. The `with` statement in Python file handling is preferred because it _____.

(A) Makes the code run faster

(B) Automatically closes the file

(C) Allows multiple files to open

(D) Encrypts the file data

Q44. Which layer of the OSI model handles the physical transmission of bits over a medium?

(A) Data Link Layer

(B) Network Layer

(C) Physical Layer

(D) Transport Layer

Q45. The protocol used for sending electronic mail (email) is:

(A) FTP

(B) SMTP

(C) HTTP

(D) SNMP

Q46. What is a ”Firewall” primarily used for in a network?

(A) Speeding up the internet

(B) Monitoring and controlling network traffic

(C) Creating backup files

(D) Connecting different topologies

Q47. Which type of malware appears to be useful software but performs malicious actions in the background?

(A) Worm

(B) Trojan Horse

(C) Adware

(D) Ransomware

Q48. In a _____ network, every node has a dedicated point-to-point link to every other node.

(A) Star

(B) Mesh

(C) Ring

(D) Bus

Q49. Which device is used to connect multiple networks that use different protocols?

(A) Hub

(B) Switch

(C) Bridge

(D) Gateway

Q50. What does the 'S' in HTTPS stand for?

(A) Simple

(B) Standard

(C) Secure

(D) System

Detailed Solutions

Q1.

Solution

Concept: The `TRUNCATE()` function in SQL is used to reduce a numeric value to a specific number of decimal places. Unlike the `ROUND()` function, which follows mathematical rules to increment the last digit if the following digit is 5 or higher, `TRUNCATE()` simply "cuts off" the digits beyond the specified limit without any adjustment to the remaining numbers.

Solution:

1. **Analyze the Input:** The query is `SELECT TRUNCATE(275.864, 1)`. The number is 275.864 and the specified decimal places is 1.
2. **Identify the Cut-off Point:** The first decimal place (tenths position) is 8. Everything after this position is subject to removal.
3. **Compare with Rounding:** If this were a `ROUND()` operation, we would look at the second decimal place (6). Since $6 \geq 5$, a round operation would result in 275.9.
4. **Apply Truncation Logic:** In truncation, we ignore the subsequent digits entirely. The digits 6 and 4 are simply discarded.
5. **Resulting Value:** After removing everything after the first decimal place, we are left with 275.8.
6. **Contextual Usage:** `TRUNCATE()` is often used in financial applications where you want to ensure a value does not exceed a certain precision without "inflating" it through rounding (e.g., calculating interest where you only pay out to the nearest cent).
7. **Conclusion:** The output is strictly 275.8 because no rounding up occurs.

Final Answer: The output is 275.8.

Answer: (B)

Q2.

Solution

Concept: String data often contains unnecessary whitespace due to user input errors or data migration issues. SQL provides ”Trim” functions to sanitize these strings. Whitespace can exist at the start of a string (leading), at the end (trailing), or both. Handling these is crucial for accurate string comparisons and clean reporting.

Solution:

1. **Evaluate Function Options:**

- `LTRIM()`: This function removes spaces only from the **Left** (leading) side of the string.
- `RTRIM()`: This function removes spaces only from the **Right** (trailing) side of the string.
- `TRIM()`: This is the comprehensive function that removes spaces from **both** the beginning and the end of the string simultaneously.

2. **Analyze Incorrect Keywords:** Keywords like `CLEAN()`, `STRIP()`, or `REMOVE()` might exist in other languages (like Python's `.strip()`), but they are not standard SQL aggregate or scalar functions for whitespace management.

3. **Syntax Example:** `SELECT TRIM(' Hello ');` would return `'Hello'`.

4. **Extended Functionality:** In modern SQL (like MySQL or PostgreSQL), `TRIM()` can also be used to remove specific characters other than spaces using the syntax `TRIM(BOTH 'x' FROM 'xxxHelloxxx')`.

5. **Conclusion:** Since the question asks for the removal of both leading and trailing spaces, the `TRIM()` function is the standard and correct choice.

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

Answer: (C)

Q3.

Solution

Concept: The `POW()` or `POWER()` function in SQL is a mathematical function used to calculate the value of a number raised to the power of another number. The syntax is typically `POW(base, exponent)`, where the base is the number to be multiplied and the exponent is the number of times to multiply it.

Solution:

1. **Identify the Arguments:** In `SELECT POW(4, 3);`, the base is 4 and the exponent is 3.
2. **Mathematical Setup:** This translates to the mathematical expression 4^3.
3. **Step-by-Step Calculation:**

- Step 1: $4 \times 4 = 16$
- Step 2: $16 \times 4 = 64$

4. **Evaluate Options:**

- Option A (12) is the result of 4×3, which is a common mistake where the base and exponent are multiplied instead of raised.
- Option B (16) is the result of 4^2.
- Option C (64) matches our calculation $(4 \times 4 \times 4)$.
- Option D (81) is the result of 3^4.

5. **Datatype Note:** The output is usually returned as a float or a double because exponents can often result in very large numbers or fractional values if the exponent is negative.
6. **Conclusion:** 4 raised to the power of 3 is 64.

Final Answer: The result is 64.

Answer: (C)

Q4.

Solution

Concept: SQL offers various functions to extract specific components from a Date or Datetime value. While `MONTH()` returns the numerical value (1−12), the `MONTHNAME()` function is designed to return the full name of the month as a string, making it highly useful for creating readable user interfaces and reports.

Solution:

1. **Analyze the Input:** The date provided is `'2026-08-15'`.

2. **Extract the Month Component:** In the `YYYY-MM-DD` format, the month is represented by the middle digits, which are `08`.

3. **Determine the Month Name:**

- 01: January, 02: February, 03: March, 04: April, 05: May, 06: June, 07: July, **08: August**.

4. **Function Specifics:** Unlike `MONTH()`, which would return the integer 8, `MONTHNAME()` returns the textual representation of that month in the current language setting of the database (defaulting to English).

5. **Compare with Related Functions:**

- `DAYNAME()` would return `'Saturday'` for this specific date.
- `DAYOFMONTH()` would return 15.
- `YEAR()` would return 2026.

6. **Conclusion:** The call to `MONTHNAME()` for the 8^{th} month of the year will return the string `'August'`.

Final Answer: The function will return August.

Answer: (B)

Q5.

Solution

Concept: Calculating the size of data is a fundamental task in SQL. However, beginners often confuse ”Aggregate functions” (which work on multiple rows) with ”Scalar functions” (which work on a single value). Finding the number of characters in a string is a scalar operation that measures the length of the character sequence.

Solution:

1. **Evaluate `LENGTH()`:** This is the standard SQL function used to return the length of a string in bytes or characters (depending on the specific SQL dialect). In most common systems like MySQL, `LENGTH('DATABASE')` will return 8 because there are 8 letters in the word.
2. **Evaluate `COUNT()`:** This is an aggregate function used to count the number of *rows* in a result set or non-null values in a column. It cannot be used to count characters inside a single string literal.
3. **Evaluate `SIZE()`:** This keyword is often used in programming (like Python's `sys.getsizeof()` or C++ vectors), but it is not a standard SQL function for string measurement.
4. **Evaluate `WIDTH()`:** This is not a standard SQL function.
5. **Detailed Logic:** The string `'DATABASE'` has: D(1), A(2), T(3), A(4), B(5), A(6), S(7), E(8). Total = 8 characters.
6. **Note on `CHAR_LENGTH()`:** Some systems also use `CHAR_LENGTH()`, which is specifically for character counts in multi-byte character sets, but `LENGTH()` is the most universally accepted answer in the CUET context.

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

Answer: (C)

Q6.

Solution

Concept: The `MID()` function in SQL is a synonym for `SUBSTR()` or `SUBSTRING()`. It is used to extract a specific portion of a string based on a starting position and a specified length. This is a common operation in data cleaning and text processing where only a specific part of a field is required.

Solution:

1. **Analyze the Arguments:** The function is `MID('Global Warming', 8, 4)`. - The source string is `'Global Warming'`. - The starting position is `8`. - The number of characters to extract is `4`.
2. **Identify the Starting Position:** In SQL, indexing starts at 1. Let's map the string: G(1), l(2), o(3), b(4), a(5), l(6), [Space](7), **W(8)**, a(9), r(10), m(11), i(12), n(13), g(14).
3. **Extract the Length:** Starting from the 8^{th} character ('W'), we take 4 characters: - 1st: W - 2nd: a - 3rd: r - 4th: m
4. **Form the Result:** Combining these four characters gives us `'Warm'`.
5. **Comparison:** If the starting position had been 1, it would return 'Glob'. If the length was omitted, it would return everything from 'Warming' to the end.
6. **Conclusion:** The extracted substring is `'Warm'`.

Final Answer: The output is 'Warm'.

Answer: (A)

Q7.

Solution

Concept: Databases track time with high precision. However, many business requirements only necessitate the date (year, month, and day) rather than the exact second. SQL provides specialized functions to filter out the time component from the system clock.

Solution:

1. **Analyze `CURDATE()`:** This function (Current Date) returns the current date in `'YYYY-MM-DD'` format. It does not include any time information (hours, minutes, seconds).
2. **Analyze `NOW()`:** This function returns both the current date and the current time (e.g., `'2026-05-06 14:30:05'`). While it contains the date, it does not provide "date only."
3. **Analyze `SYSTIME()`:** This function typically returns the system time only, excluding the date.
4. **Analyze `TODAY()`:** While common in some specific SQL dialects (like Informix), it is not a standard ANSI SQL or MySQL function. `CURDATE()` or `CURRENT_DATE` are the standard identifiers.
5. **Contextual Usage:** `CURDATE()` is frequently used in `WHERE` clauses to filter records from "today" without having to handle the varying seconds of the `NOW()` function.
6. **Conclusion:** For displaying only the date part of the system's current temporal data, `CURDATE()` is the correct function.

Final Answer: Use the `CURDATE()` function.

Answer: (B)

Q8.

Solution

Concept: The `ROUND()` function can take a negative second argument. While a positive number rounds to the right of the decimal point, a negative number rounds to the left. A value of -1 rounds the number to the nearest 10, -2 to the nearest 100, and so on.

Solution:

1. **Identify the Goal:** We need to round 1234.5678 to the -1 position (the tens place).
2. **Locate the Digits:** - The tens digit is 3. - The digit to its immediate right (the units place) is 4.
3. **Apply Rounding Rules:** Since the units digit (4) is less than 5, we do not increment the tens digit. We simply turn all digits to the right of the tens place into zeros and remove the decimal part.
4. **Execute Calculation:** - The 123 stays largely the same, but the 4 becomes 0. - Result: 1230.
5. **Verify with other values:** - If the number was 1235.0, the result would be 1240. - If the argument was -2, the result would be 1200.
6. **Conclusion:** Rounding 1234.5678 to the nearest ten results in 1230.

Final Answer: The result is 1230.

Answer: (A)

Q9.

Solution

Concept: Relational Algebra includes several ”Set” operators. These operators work on two relations that are ”Union Compatible”—meaning they have the same number of attributes and matching domains for those attributes. The Union operator is a fundamental tool for merging data from different sources.

Solution:

1. **Analyze Union ($\cup$):** The Union operator combines all tuples from Relation A and Relation B into a single result. Following set theory, any tuple that exists in both relations is included only once in the result to eliminate redundancy.
2. **Analyze Intersection ($\cap$):** This operator returns only the tuples that appear in **both** relations.
3. **Analyze Set Difference ($-$):** This returns tuples that are in the first relation but **not** in the second.
4. **Analyze Cartesian Product ($\times$):** This combines every row of the first table with every row of the second. It does not require union compatibility and usually results in a much larger relation.
5. **Check the Condition:** The question specifies ”combining all rows” and ”removing duplicates.” This is the exact definition of the Union ($\cup$) operator in relational algebra.
6. **Conclusion:** To achieve a merged list without repetitions, the Union operator is used.

Final Answer: The operator is Union ($\cup$).

Answer: (B)

Q10.

Solution

Concept: Database design involves identifying various keys to ensure uniqueness. A ”Candidate Key” is any attribute or set of attributes that can uniquely identify a row. A table can have multiple Candidate Keys. The selection process for the Primary Key leaves the remaining keys with a specific designation.

Solution:

1. **Identify the Pool:** Imagine a ’Student’ table with ’Roll_No’ and ’Email’. Both are unique and non-null. Both are **Candidate Keys**.
2. **Selection:** The Database Administrator chooses ’Roll_No’ to be the **Primary Key**.
3. **The Remaining Key:** The ’Email’ attribute, which was a ”candidate” but didn’t win the ”primary” spot, is now called an **Alternate Key**.
4. **Evaluate Other Terms:** - **Secondary Key:** This usually refers to a field used for indexing/searching but doesn’t have to be unique. - **Foreign Key:** A key used to link two tables together. - **Composite Key:** A key that consists of more than one attribute.
5. **Conclusion:** By definition, an Alternate Key is simply any candidate key that is not currently functioning as the primary key.

Final Answer: It is called an Alternate Key.

Answer: (C)

Q11.

Solution

Concept: In the relational model, every table (relation) has two fundamental properties that describe its size: Cardinality and Degree. These terms are often confused by students. Cardinality refers to the horizontal size (number of rows), while Degree refers to the vertical size (number of columns).

Solution:

1. **Define Attributes:** In database terminology, an ”attribute” is a named column in a relation. It represents a specific property of the entity being modeled (e.g., Name, Age, or ID).

2. **Analyze the Term ’Degree’:** The degree of a relation is defined as the total number of attributes (columns) it contains. For example, if a table has columns for ’ID’, ’Name’, and ’GPA’, its degree is 3.

3. **Analyze ’Cardinality’:** This refers to the number of tuples (rows) currently stored in the table. While the degree is usually fixed by the table’s schema, the cardinality changes as records are inserted or deleted.

4. **Evaluate Options:**

- **(A) Number of Tuples:** This is Cardinality.
- **(B) Number of Attributes:** This is the correct definition of Degree.
- **(C) Number of Constraints:** Constraints (like NOT NULL) do not define the degree.
- **(D) Number of Keys:** A table can have many keys, but this is independent of the degree count.

5. **Conclusion:** The degree represents the vertical dimension of the table, which is equivalent to the number of attributes.

Final Answer: The degree refers to the Number of Attributes.

Answer: (B)

Q12.

Solution

Concept: To ensure data integrity, every database transaction must follow the ACID properties: Atomicity, Consistency, Isolation, and Durability. These properties ensure that even in the event of errors, power failures, or concurrent access, the database remains in a valid state.

Solution:

1. **Analyze Atomicity:** This property follows the ”All or Nothing” rule. A transaction might involve multiple steps (e.g., deducting money from Account A and adding it to Account B). Atomicity ensures that if any single part of the transaction fails, the entire transaction is aborted, and the database is rolled back to its previous state.
2. **Analyze Consistency:** This ensures that a transaction takes the database from one valid state to another, maintaining all predefined rules (constraints).
3. **Analyze Isolation:** This ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.
4. **Analyze Durability:** This ensures that once a transaction has been committed, it will remain so, even in the event of a system failure (e.g., the data is saved to non-volatile memory).
5. **Match the Definition:** The question asks about a transaction being an ”indivisible unit.” Indivisibility is the literal meaning of the word ”Atomic.”
6. **Conclusion:** Therefore, the property that treats a transaction as a single unit is Atomicity.

Final Answer: The property is Atomicity.

Answer: (C)

Q13.

Solution

Concept: The physical topology of a network describes how nodes are connected via cabling. The Bus Topology was one of the earliest and simplest forms of local area networking, characterized by a shared communication line.

Solution:

1. **Analyze Bus Topology:** In a bus network, all workstations, servers, and peripherals are connected to a single, continuous central cable called the ”backbone” or ”bus.” Data travels along the cable, and each node checks the destination address to see if the data is meant for them.
2. **Analyze Star Topology:** Here, every node connects to a central hub or switch via its own dedicated cable. There is no single ”central cable” shared by all nodes.
3. **Analyze Ring Topology:** Nodes are connected in a closed loop. Each node acts as a repeater to pass the signal to the next node.
4. **Analyze Mesh Topology:** Every node is connected to every other node, requiring massive amounts of cabling and no central backbone.
5. **Hardware Detail:** Bus topologies typically use coaxial cables and ”T-connectors” to attach devices to the backbone. ”Terminators” are required at both ends of the central cable to prevent signal bounce.
6. **Conclusion:** The description of connecting all computers to a ”single central cable” is the unique identifier of the Bus Topology.

Final Answer: The topology is Bus Topology.

Answer: (B)

Q14.

Solution

Concept: For devices to communicate on an IP network, each must have a unique IP address. These can be assigned manually (static) or automatically (dynamic). The Dynamic Host Configuration Protocol (DHCP) is the standard mechanism for automated IP management.

Solution:

1. **Analyze DHCP:** DHCP is a network management protocol used on UDP/IP networks. A DHCP server dynamically assigns an IP address and other network configuration parameters (like subnet mask and default gateway) to each device on a network so they can communicate with other IP networks.
2. **Evaluate DNS:** The Domain Name System translates human-readable names (like google.com) into IP addresses. It does not assign IP addresses to devices.
3. **Evaluate HTTP:** This is used for transferring web content; it has nothing to do with network layer addressing.
4. **Evaluate SMTP:** This is used for sending emails.
5. **Operational Logic:** When a device (like a smartphone) joins a Wi-Fi network, it sends a broadcast "DHCP Discover" message. The router's DHCP server responds with an "Offer" of an IP address from a pool, simplifying network administration.
6. **Conclusion:** DHCP is the protocol responsible for the dynamic and automatic assignment of IP addresses.

Final Answer: The protocol is DHCP.

Answer: (C)

Q15.

Solution

Concept: Internet Protocol version 4 (IPv4) is the fourth version of the Internet Protocol and the first version to be widely deployed. It uses a logical addressing scheme that defines the size and structure of addresses used to identify devices on a network.

Solution:

1. **Analyze IPv4 Structure:** An IPv4 address is typically represented in ”dotted-decimal” notation (e.g., 192.168.1.1).
2. **Count the Octets:** The address consists of four decimal numbers, each called an ”octet.” Each octet represents 8 bits (1 byte).
3. **Calculate Total Bits:** 4 octets $\times$ 8 bits/octet = 32 bits.
4. **Address Space:** With 32 bits, the total number of unique possible addresses is 2^{32}, which is approximately 4.3 billion.
5. **Compare with IPv6:** Due to the exhaustion of IPv4 addresses, IPv6 was developed. IPv6 uses 128 bits, allowing for a vastly larger number of addresses.
6. **Evaluate Options:**

- 16 bits: Used for port numbers.
- 48 bits: Used for MAC (hardware) addresses.
- 128 bits: Used for IPv6 addresses.

7. **Conclusion:** The standard bit-length for an IPv4 address is exactly 32 bits.

Final Answer: An IPv4 address is composed of 32 bits.

Answer: (B)

Q16.

Solution

Concept: The Stack is an Abstract Data Type (ADT) that follows the LIFO (Last-In, First-Out) principle. While `Push` and `Pop` are the primary operations for modifying the stack, the `Peek` (sometimes called `Top`) operation is a non-destructive query used to examine the state of the stack.

Solution:

1. **Analyze Push:** This operation adds an element to the top of the stack. It modifies the stack and increases its size.
2. **Analyze Pop:** This operation removes the element from the top of the stack and typically returns it. It modifies the stack and decreases its size.
3. **Analyze Peek:** The `Peek` operation allows the program to look at the value of the topmost element without removing it. The stack remains unchanged. This is useful in algorithms where you need to check if the next character (like a parenthesis) matches a certain condition before deciding to pop.
4. **Analyze Overflow:** This is not an operation but a "condition" or "error state" that occurs when you try to `Push` an element onto a stack that has already reached its maximum memory capacity.
5. **Conclusion:** Since the question asks for an operation that checks the top element without removal, `Peek` is the correct technical term.

Final Answer: The operation is `Peek`.

Answer: (C)

Q17.

Solution

Concept: Postfix notation (Reverse Polish Notation) is an expression format where operators follow their operands. To convert an Infix expression ($Operand1$ Operator $Operand2$) to Postfix, we must respect the Order of Precedence (BODMAS/PEMDAS). Multiplication (*) has higher precedence than Addition (+).

Solution:

1. **Analyze the Infix Expression:** $A + B * C$
2. **Identify Precedence:** The multiplication ($B * C$) must be evaluated before the addition.
3. **Step 1 (Convert Multiplication):** Treat B and C as operands for the $*$ operator. In postfix, this becomes $BC*$.
4. **Step 2 (Substitute):** Now the expression looks like $A + [BC*]$.
5. **Step 3 (Convert Addition):** Treat A as the first operand and the result $[BC*]$ as the second operand. The operator is +. In postfix, the operator moves to the end of its operands.
6. **Result:** $ABC*+$
7. **Comparison with other options:** - $AB + C*$ would correspond to $(A + B) * C$. - $+A * BC$ is the Prefix form of the expression.
8. **Conclusion:** Following the rules of operator precedence, the correct postfix string is $ABC*+$.

Final Answer: The postfix form is $ABC*+$.

Answer: (B)

Q18.

Solution

Concept: Linear data structures are categorized by how they allow access to their elements. While a Linked List or Array might allow access to any index, specific ADTs like Stacks and Queues restrict access to the ends of the structure to enforce specific logical behaviors.

Solution:

1. **Analyze Stack Behavior:** A stack is a "single-ended" structure. Both insertion (`Push`) and deletion (`Pop`) occur at the same end, known as the "Top." This creates the LIFO behavior.
2. **Analyze Queue Behavior:** A queue is "double-ended" in terms of logic. Elements are added at the "Rear" and removed from the "Front" (FIFO).
3. **Analyze Linked List:** A Linked List is a general structure that can be used to implement both stacks and queues, but it does not inherently restrict access to one end.
4. **Analyze Tree:** A tree is a non-linear data structure.
5. **Match the Definition:** The question specifies a structure where elements are added and removed from the **same end**. This is the fundamental definition of a Stack.
6. **Real-world Analogy:** Think of a stack of trays in a cafeteria. You place a new tray on the top and take the next tray from the top. You do not add to the bottom and remove from the top.

Final Answer: The data structure is a Stack.

Answer: (C)

Q19.

Solution

Concept: Python lists are often used to implement other data structures. The `pop()` method in Python accepts an index as an argument. Understanding which index corresponds to which end of a conceptual data structure is vital for implementation questions in the CUET-UG exam.

Solution:

1. **Analyze `L.pop()`:** Without an argument, `pop()` removes the last element (index -1). This is efficient ($O(1)$) and is used for Stacks.
2. **Analyze `L.pop(0)`:** This explicitly tells Python to remove the element at index 0.
3. **Identify Index 0:** In a Python list, index 0 always represents the very first element.
4. **Relate to Queues:** In a Queue (FIFO), the element that was added first is the one that should be removed first. If we use `append()` to add elements to the end, then the "Front" of the queue is at index 0.
5. **Complexity Note:** While `L.pop(0)` removes the first element, it is an $O(n)$ operation because all subsequent elements must be shifted one position to the left. This is why `collections.deque` is preferred for large queues in Python.
6. **Conclusion:** Executing `pop(0)` results in the removal of the first element of the list.

Final Answer: The first element is removed.

Answer: (B)

Q20.

Solution

Concept: Evaluating a Postfix expression involves using a stack. We scan from left to right. Operands are pushed onto the stack. When an operator is encountered, we pop the required number of operands (two for binary operators), perform the operation, and push the result back.

Solution:

1. **Expression:** $8, 4, /, 2, +$
2. **Scan 8:** Operand. Push onto stack. [Stack: 8]
3. **Scan 4:** Operand. Push onto stack. [Stack: 8, 4]
4. **Scan /:** Operator. Pop 4 (Operand 2) and 8 (Operand 1). Perform $8/4 = 2$. Push 2. [Stack: 2]
5. **Scan 2:** Operand. Push onto stack. [Stack: 2, 2]
6. **Scan +:** Operator. Pop 2 (Operand 2) and 2 (Operand 1). Perform $2 + 2 = 4$. Push 4. [Stack: 4]
7. **Final Result:** The scanning is complete, and the final value remaining on the stack is 4.
8. **Verification:** In infix, this would be $(8/4) + 2$, which is $2 + 2 = 4$.
9. **Common Error:** Be careful with non-commutative operators like division or subtraction. The first element popped is the right-hand operand, and the second element popped is the left-hand operand.

Final Answer: The result is 4.

Answer: (A)

Q21.

Solution

Concept: The ”Undo” mechanism is one of the most classic real-world applications of the Stack data structure. It relies on the LIFO (Last-In, First-Out) principle to revert the state of an application. By storing each user action as an object or a state on a stack, the application can easily backtrack through history in the exact reverse order in which the actions were performed.

Solution:

1. **Analyze the Requirement:** An ”Undo” feature requires the system to remember a history of actions. When the user clicks ”Undo,” the very *last* action performed must be the *first* one to be reversed.
2. **Evaluate Stack (LIFO):** As a user types or deletes text, each discrete action is ”Pushed” onto the Undo Stack. When ”Undo” is triggered, the most recent action is ”Popped” from the stack and its inverse is executed. This perfectly matches the LIFO behavior.
3. **Evaluate Queue (FIFO):** If we used a Queue, the ”Undo” button would reverse the *first* action the user ever took (e.g., the first letter typed an hour ago), which is the opposite of the desired behavior.
4. **Redo Implementation:** Typically, software uses two stacks. When an action is popped from the ”Undo Stack,” it is pushed onto a ”Redo Stack.”
5. **Conclusion:** Because of the necessity to access the most recent data point first, the Stack is the only logical choice for an undo/redo architectural pattern.

Final Answer: The Stack is the most suitable data structure.

Answer: (B)

Q22.

Solution

Concept: In the study of data structures, boundary conditions are critical for robust programming. These conditions occur when an operation is performed that violates the current state or capacity of the structure. For linear structures like Stacks, two primary boundary errors exist: Overflow and Underflow.

Solution:

1. **Analyze Overflow:** This occurs when the stack has a fixed size (Static Stack) and the program attempts to `Push` an element when the stack is already full.

2. **Analyze Underflow:** This occurs when the stack is empty (size is 0) and the program attempts to `Pop` an element or `Peek` at the top. Since there is no data to retrieve, the operation is invalid.

3. **Logical Check in Code:** A well-written `pop()` function should always check `if (isEmpty())` before proceeding. If this check is missing, the program might crash or return "garbage" values.

4. **Evaluate Options:** - **Garbage Collection:** This is a memory management process, not a stack state. - **Emptyflow:** This is not a standard technical term in computer science.

5. **Conclusion:** The specific term for trying to remove an item from an empty stack is "Underflow."

Final Answer: This situation is known as Underflow.

Answer: (B)

Q23.

Solution

Concept: A Queue is a linear data structure that manages data using a First-In, First-Out (FIFO) methodology. To maintain this logic effectively, a Queue must track two distinct points of the collection to ensure that insertions and deletions happen at opposite ends.

Solution:

1. **The 'Front' Pointer:** This pointer tracks the head of the queue. Its primary role is to identify the location of the element that has been in the queue the longest. All deletion (`Dequeue`) operations occur at the `Front`.

2. **The 'Rear' Pointer:** This pointer (sometimes called 'Back' or 'Tail') tracks the end of the queue. Its role is to identify where the next new element should be placed. All insertion (`Enqueue`) operations occur at the `Rear`.

3. **Initialization:** When a queue is empty, both pointers are typically initialized to -1 or `None`.

4. **Compare with Stack:** A Stack only requires 1 pointer (the `Top`) because all activity happens at one end. A Queue, by its dual-ended nature, necessitates 2 pointers.

5. **Conclusion:** Managing a linear queue requires exactly two pointers to maintain the FIFO integrity.

Final Answer: 2 pointers are primarily required.

Answer: (B)

Q24.

Solution

Concept: Data structures are broadly classified into two categories based on how their elements are arranged: Linear and Non-linear. Linear structures form a sequence where each element (except the first and last) has a unique predecessor and successor. Non-linear structures represent complex relationships, such as hierarchies or networks.

Solution:

1. **Evaluate Stack, Queue, and List:** These are all linear data structures. Whether implemented via arrays or linked nodes, the data follows a straight, sequential path.
2. **Analyze the Tree:** A Tree is a non-linear data structure. It consists of nodes connected in a parent-child relationship. A single node (the parent) can have multiple successors (children), creating a branching, hierarchical structure.
3. **Other Non-linear Examples:** Graphs are another major category of non-linear structures, where nodes can be connected in any arbitrary pattern (cycles, networks, etc.).
4. **Why the distinction matters:** Linear structures are usually traversed in $O(n)$ time. Non-linear structures like Binary Search Trees allow for much faster searching ($O(\log n)$) because of their non-sequential arrangement.
5. **Conclusion:** Among the choices, the Tree is the only structure that does not store data in a linear sequence.

Final Answer: The Tree is a non-linear data structure.

Answer: (C)

Q25.

Solution

Concept: Converting from Prefix notation to Infix notation requires identifying the operators and their associated operands by scanning the expression. Since Prefix places the operator *before* the operands ($+AB$), the conversion logic involves finding the operator and wrapping it between its next two operands.

Solution:

1. **Analyze the Expression:** $+ * ABC$
2. **Identify the Structure:** We see two operators ($+, *$) followed by three operands (A, B, C).
3. **Step 1 (Inner Operation):** Look for the operator closest to the operands. Here, $*$ is followed by A and B. - Convert to infix: $(A * B)$.
4. **Step 2 (Outer Operation):** Substitute the result back into the expression: $+(A * B)C$.
5. **Final Conversion:** Now treat $(A * B)$ as the first operand and C as the second operand for the $+$ operator. - Move the $+$ between them: $(A * B) + C$.
6. **Precedence Check:** In mathematical notation, $A * B + C$ is evaluated as $(A * B) + C$ even without parentheses because multiplication has higher precedence. Thus, both $(A * B) + C$ and $A * B + C$ are correct infix representations.
7. **Conclusion:** Option (B) provides the clearest infix representation of the prefix string $+ * ABC$.

Final Answer: The infix expression is $(A * B) + C$.

Answer: (B)

Q26.

Solution

Concept: Abstract Data Types (ADTs) define a set of operations (like Enqueue and Dequeue), but they don't dictate how those operations are implemented. In Python, lists are the most common tool for implementing these structures. To maintain the First-In, First-Out (FIFO) property of a Queue, we must consistently add to one end and remove from the other.

Solution:

1. **Identify the Enqueue Operation:** This operation adds an element to the "Rear" of the queue.
2. **Evaluate `append()`:** The `list.append(x)` method adds an element `x` to the very end of the list. If we treat the end of the list as the "Rear," then `append()` is the perfect implementation of `Enqueue`. It is also highly efficient ($O(1)$ amortized).
3. **Evaluate `insert()`:** While `insert(0, x)` could add an element to the front, it is an $O(n)$ operation and is less commonly used as the "Enqueue" step when `append` is available.
4. **Evaluate `remove()`:** This method removes a specific *value*, not a position. It is not used for standard queue operations.
5. **Consistency:** If `append()` is used for Enqueue, then the Dequeue operation must be `pop(0)` to remove the element from the other end.
6. **Conclusion:** The `append()` method is the standard Python equivalent for adding an item to the end of a queue.

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

Answer: (C)

Q27.

Solution

Concept: Linear Search (or Sequential Search) is the most basic searching algorithm. It works by checking every element in a collection, one by one, until a match is found or the end of the collection is reached. Understanding the maximum number of operations is essential for calculating worst-case time complexity.

Solution:

1. **Analyze the Best Case:** The target element is at the very first index. Only 1 comparison is needed.
2. **Analyze the Average Case:** On average, the algorithm will check $n/2$ elements.
3. **Analyze the Worst Case:** The worst-case scenario occurs when: - The target element is at the very last position (index $n-1$). - The target element is not present in the list at all.
4. **Calculate for $n = 100$:** In either worst-case scenario, the algorithm must compare the target against every single one of the 100 elements before it can conclude the search.
5. **Mathematical Definition:** For a list of size n, the maximum number of comparisons is n.
6. **Conclusion:** For 100 elements, the maximum number of comparisons required is exactly 100.

Final Answer: The maximum number of comparisons is 100.

Answer: (D)

Q28.

Solution

Concept: Advanced sorting algorithms often use "Divide and Conquer" to improve efficiency from $O(n^2)$ to $O(n \log n)$. This approach involves breaking a large problem into smaller sub-problems, solving them, and combining the results.

Solution:

1. **Analyze Bubble, Selection, and Insertion Sort:** These are simple $O(n^2)$ algorithms. They do not use divide and conquer; they process elements one by one or through repeated passes.
2. **Analyze Quick Sort:** This is a classic divide-and-conquer algorithm. It works by selecting a "pivot" element from the array.
3. **The Partitioning Step:** The array is reordered so that all elements smaller than the pivot go to its left, and all elements larger go to its right. The pivot is then in its final sorted position.
4. **Recursive Call:** The algorithm then recursively applies the same logic to the left sub-array and the right sub-array.
5. **Performance:** Quick Sort is generally faster in practice than Merge Sort (another divide-and-conquer algorithm) because it sorts "in-place" without needing extra memory.
6. **Conclusion:** The use of a "pivot" is the defining characteristic of Quick Sort.

Final Answer: The algorithm is Quick Sort.

Answer: (C)

Q29.

Solution

Concept: Bubble Sort operates by comparing adjacent elements and swapping them if they are in the wrong order (ascending order usually means $Left > Right \implies Swap$). In one "pass," the algorithm travels from the start to the end of the unsorted part of the list.

Solution:

1. **Initial List:** $[5, 1, 4, 2, 8]$
2. **Pass 1, Step 1:** Compare 5 and 1. $5 > 1$, so swap. $\rightarrow [1, 5, 4, 2, 8]$
3. **Pass 1, Step 2:** Compare 5 and 4. $5 > 4$, so swap. $\rightarrow [1, 4, 5, 2, 8]$
4. **Pass 1, Step 3:** Compare 5 and 2. $5 > 2$, so swap. $\rightarrow [1, 4, 2, 5, 8]$
5. **Pass 1, Step 4:** Compare 5 and 8. $5 < 8$, so no swap. $\rightarrow [1, 4, 2, 5, 8]$
6. **Result of Pass 1:** The largest element (8) has "bubbled up" to the end. The state of the list is now $[1, 4, 2, 5, 8]$.
7. **Observation:** Note that even though 5 is now in its correct final spot as well, the "official" result of the first pass is specifically the state after the final comparison of that traversal.
8. **Conclusion:** Option (A) correctly represents the list after the first complete pass.

Final Answer: The list will be $[1, 4, 2, 5, 8]$.

Answer: (A)

Q30.

Solution

Concept: Time complexity analysis helps identify which algorithms become inefficient as data grows. $O(n^2)$ complexity implies that if the input size doubles, the time taken quadruples. This is typical of algorithms that use "nested loops" (a loop inside a loop).

Solution:

1. **Evaluate Binary Search:** This is a very efficient $O(\log n)$ algorithm. It is never $O(n^2)$.
2. **Evaluate Linear Search:** This is an $O(n)$ algorithm. It only checks each element once.
3. **Evaluate Bubble Sort:** Bubble Sort uses an outer loop to control the number of passes $(n-1)$ and an inner loop to perform the comparisons $(n-1, n-2, \ldots)$. In the worst case (a reverse-sorted list), it performs roughly $n^2/2$ comparisons. In Big O notation, we drop constants, making it $O(n^2)$.
4. **Selection Sort and Insertion Sort:** These also share the $O(n^2)$ worst-case complexity due to their nested-loop structures.
5. **Conclusion:** Out of the provided options, Bubble Sort is the only one that degrades to $O(n^2)$ performance.

Final Answer: Bubble Sort has a worst-case complexity of $O(n^2)$.

Answer: (B)

Q31.

Solution

Concept: Selection Sort is a comparison-based sorting algorithm. It is known for its simplicity and the fact that it performs a predictable number of swaps. Unlike Bubble Sort, which may swap elements many times during a single pass, Selection Sort only performs a maximum of one swap per pass.

Solution:

1. **Analyze the Algorithm:** In each pass of Selection Sort, the algorithm finds the minimum element from the unsorted portion of the list and swaps it with the element at the beginning of that unsorted portion.
2. **Calculate Swaps per Pass:** Since we only perform a swap once we have found the absolute minimum for that pass, we perform exactly 1 swap (or 0 if the minimum is already in place) per pass.
3. **Total Passes:** For a list of size n, the algorithm requires $n-1$ passes to fully sort the data.
4. **Total Swaps:** Therefore, the total number of swaps in the worst case is $n-1$.
5. **Comparison with Comparisons:** While the number of *comparisons* is $O(n^2)$, the number of *swaps* is $O(n)$. This makes Selection Sort more efficient than Bubble Sort in scenarios where the "cost" of moving data (swapping) is very high compared to the cost of comparing it.
6. **Conclusion:** In a list of n elements, the maximum number of times elements are exchanged is $n-1$.

Final Answer: Selection Sort performs $n-1$ swaps in the worst case.

Answer: (C)

Q32.

Solution

Concept: Algorithm efficiency is usually measured by how the time taken scales with the input size. For searching, we compare Linear Search ($O(n)$) and Binary Search ($O(\log n)$). As the size of the dataset reaches the millions, the difference in efficiency becomes staggering.

Solution:

1. **Analyze Linear Search:** For $1,000,000$ elements, Linear Search would require, on average, $500,000$ comparisons and a maximum of $1,000,000$ comparisons if the element is at the end or missing.
2. **Analyze Binary Search:** Binary Search repeatedly halves the search space. To find the number of comparisons, we solve $2^k \approx 1,000,000$. - $2^{10} \approx 1,000$ - $2^{20} \approx 1,000,000$
3. **Compare the Results:** - Linear Search: Up to $1,000,000$ steps. - Binary Search: Approximately 20 steps.
4. **The Prerequisite:** The question specifies a **sorted list**. This allows us to use Binary Search. If the list were unsorted, we would be forced to use the slower Linear Search.
5. **Conclusion:** Binary Search is significantly faster for large, sorted datasets because its complexity is logarithmic rather than linear.

Final Answer: Binary Search is faster for a sorted list of 1,000,000 elements.

Answer: (B)

Q33.

Solution

Concept: The "Best Case" of an algorithm is the scenario where it performs the minimum amount of work possible. For searching algorithms, this usually occurs when the target element is found immediately upon the very first operation.

Solution:

1. **Analyze Linear Search Logic:** The algorithm starts at the first index (index 0) and moves sequentially toward the end.
2. **Identify the Minimum Work:** If the target element X is located at index 0, the very first comparison `if (List[0] == X)` will be true.
3. **Complexity in Best Case:** In this specific instance, the algorithm performs exactly 1 comparison. This is represented as $O(1)$ or "Constant Time" complexity.
4. **Contrast with Worst Case:** The worst case occurs when the element is at the last position or not present, requiring n comparisons ($O(n)$).
5. **Evaluate Options:** - (A) End: Worst case (n comparisons). - (B) Not in list: Worst case (n comparisons). - (C) First position: Best case (1 comparison).
6. **Conclusion:** Linear search is at its most efficient when the desired item is the very first one checked.

Final Answer: The best-case scenario is when the element is at the first position.

Answer: (C)

Q34.

Solution

Concept: Sorting algorithms are often distinguished by how they handle the unsorted portion of the data. Insertion Sort is unique because it treats the list as a growing ”sorted” collection, inserting one new element into its proper place during each iteration.

Solution:

1. **Analyze Insertion Sort:** This algorithm takes the first element of the ”unsorted” sub-array and shifts it left into the ”sorted” sub-array until it reaches its correct relative position. This is the logic described in the question.
2. **Analyze Selection Sort:** This algorithm scans the *entire* unsorted part to find the absolute minimum before making a move. It doesn't just ”pick the first element” of the unsorted part; it picks the *smallest* element.
3. **Analyze Bubble Sort:** This algorithm compares adjacent pairs and doesn't maintain a strict ”sorted/unsorted” split in the same way; instead, the sorted part grows at the end of the list.
4. **Key Distinction:** The phrase ”picks the first element of the unsorted part to place it correctly” is the definitive procedural step of Insertion Sort.
5. **Conclusion:** Because Insertion Sort processes the next available unsorted item and ”inserts” it into the already-sorted prefix, it matches the question's description.

Final Answer: The sorting algorithm is Insertion Sort.

Answer: (B)

Q35.

Solution

Concept: Standard implementations of simple sorting algorithms often have fixed nested loops that execute regardless of the initial order of the data. Unless an explicit ”early exit” optimization is added, these algorithms will always perform the same number of comparisons.

Solution:

1. **Analyze Selection Sort:** This algorithm always looks for the minimum element by comparing the current element with every other element in the unsorted portion. It has no way to ”know” the list is already sorted without completing these comparisons. Thus, it is always $O(n^2)$.
2. **Analyze Bubble Sort (Unoptimized):** Without a ”swap flag,” Bubble Sort will continue to perform $n - 1$ passes, each with multiple comparisons, even if no swaps are needed. It remains $O(n^2)$.
3. **Analyze Insertion Sort (Standard):** While Insertion Sort is often cited as $O(n)$ for sorted lists (because the ”while” loop for shifting never runs), many basic textbook implementations still count the initial comparisons. However, compared to Selection Sort, it is much more likely to be optimized.
4. **Focus of the Question:** The question asks which algorithm (without optimization) **still** takes $O(n^2)$. Since all three standard, non-optimized versions use two nested loops that iterate roughly n times each, they all result in $O(n^2)$ time complexity.
5. **Conclusion:** All the listed simple sorting algorithms have $O(n^2)$ complexity in their basic, unoptimized forms.

Final Answer: All of the above.

Answer: (D)

Q36.

Solution

Concept: Exception handling in Python allows a program to deal with errors gracefully rather than crashing. The `try` block contains code that might fail, `except` handles specific errors, and `finally` ensures that clean-up code runs regardless of the outcome. This flow of control is essential for building reliable applications.

Solution:

1. **Trace the Code:** - The list `x` has 3 elements (indices 0, 1, 2). - Inside the `try` block, the code attempts to access `x[5]`. 2. **Exception Identification:** Since index 5 does not exist, Python raises an `IndexError`. 3. **Execute the Except Block:** The program looks for a matching `except` block. The code has `except IndexError:`, so this block executes, printing `"Index Error"`. The `end=" "` parameter keeps the cursor on the same line. 4. **Evaluate the Finally Block:** The `finally` block is ”guaranteed” to execute whether an exception was raised or not. Therefore, it prints `"Done"`. 5. **Combine the Output:** The total output becomes `"Index Error Done"`. 6. **Logical Check:** The `print(x[5])` line never successfully finishes, so nothing else from the `try` block would have printed if there were more lines. 7. **Conclusion:** The combination of the caught error message and the finality message results in option (C).

Final Answer: The output is ”Index Error Done”.

Answer: (C)

Q37.

Solution

Concept: Python’s `open()` function uses ”modes” to define the permissions for a file stream. Standard modes like `'r'` (read) and `'w'` (write) are exclusive. However, ”plus” modes (+) are used to open a file for updating, meaning both reading and writing are permitted on the same file handle.

Solution:

1. **Analyze `'r'`:** Only allows reading. If the file is missing, it raises an error.
2. **Analyze `'w'`:** Only allows writing. It overwrites existing files.
3. **Analyze `'r+'`:** This opens the file for both **Reading and Writing**. The file pointer is placed at the beginning of the file. This is the correct mode for ”updating” a file.
4. **Analyze `'w+'`:** This also allows both reading and writing, but it **overwrites** the existing file first. If you want to read the existing content and then write, `'r+'` is the safer choice.
5. **Analyze `'a+'`:** This allows reading and appending. The pointer is at the end.
6. **Conclusion:** For general simultaneous read/write access to an existing file, the `'r+'` mode is the standard technical requirement.

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

Answer: (B)

Q38.

Solution

Concept: When a file is opened in a "write" or "append" mode, Python provides specific methods to send data to the disk. Understanding the difference between writing a single block of text versus multiple blocks is a common point of evaluation in the File I/O chapter.

Solution:

1. **Analyze `write()`:** This method takes exactly **one string** as an argument and writes it to the file. For example: `f.write("Hello World")`. It does not automatically add newlines.
2. **Analyze `writelines()`:** This method takes a **list of strings** (or any iterable) and writes them to the file sequentially. It is used for bulk writing.
3. **Analyze `put()` and `output()`:** These are not standard built-in methods for Python file objects. `put()` is commonly used in networking (FTP) or queues, but not basic file I/O.
4. **Functionality:** The `write()` function returns the number of characters written to the file.
5. **Conclusion:** To write a single string to a file, the `write()` method is the correct tool.

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

Answer: (A)

Q39.

Solution

Concept: While Python raises exceptions automatically when it encounters errors (like `ZeroDivisionError`), programmers can also force an exception to occur based on custom logic (e.g., if a user enters a negative age). This is known as "throwing" or "raising" an exception.

Solution:

1. **Evaluate the `raise` keyword:** In Python, the `raise` statement is used to trigger a specific exception. The syntax is `raise ExceptionName("Error Message")`.
2. **Evaluate `throw`:** This is the keyword used in languages like Java, C++, and C# to achieve the same result. It is a common "distractor" in Python exams.
3. **Evaluate `stop` and `error`:** These are not keywords for triggering exceptions in Python.
4. **Example Usage:** `if x < 0: raise ValueError("No negative numbers allowed")`
5. **Conclusion:** To manually signal that an error has occurred in Python code, the programmer must use the `raise` keyword.

Final Answer: The keyword is `raise`.

Answer: (C)

Q40.

Solution

Concept: The `pickle` module is used for serializing and de-serializing Python objects. Because these objects are converted into a stream of bytes, they cannot be handled by standard text-reading modes. They require "Binary" access modes.

Solution:

1. **Identify the Operation:** `pickle.load()` is used to read an existing serialized object from a file and convert it back into a Python object (like a list or dictionary).
2. **Determine the File Type:** Pickled files are **binary files**.
3. **Select the Mode:** - `'r'` is for Reading Text. - `'w'` is for Writing Text. - `'rb'` stands for **Read Binary**. This is required for `pickle.load()`. - `'wb'` stands for Write Binary. This is required for `pickle.dump()`.
4. **Error Prevention:** If you try to open a binary file in `'r'` mode, Python will likely raise a `UnicodeDecodeError` because it will try to interpret the raw bytes as text characters.
5. **Conclusion:** To successfully load data using the `pickle` module, the file must be opened in `'rb'` mode.

Final Answer: The file must be opened in `'rb'` mode.

Answer: (C)

Q41.

Solution

Concept: Random access in a file allows a program to skip to specific data without reading everything sequentially. The `seek()` method is the primary tool for this. It takes two arguments: `offset` (how many bytes to move) and `whence` (the reference point).

Solution:

1. **Analyze the `whence` parameter:** - `0`: Reference from the beginning of the file (default). - `1`: Reference from the current pointer position. - `2`: Reference from the end of the file.
2. **Analyze `f.seek(0, 2)`:** - The offset is `0`. - The reference point is `2` (the end of the file).
3. **Determine the Action:** This command tells the file pointer to move 0 bytes from the end of the file. Effectively, this places the cursor at the very last byte of the file.
4. **Common Use Case:** This is often done right before calling `f.tell()` to find the total size of the file in bytes, or before an append operation in some binary formats.
5. **Conclusion:** The operation moves the file pointer to the end of the file.

Final Answer: It moves the pointer to the end of the file.

Answer: (B)

Q42.

Solution

Concept: Mapping exceptions to specific errors is a core debugging skill. Python's `KeyError` is a subclass of `LookupError`. It occurs when a mapping (dictionary) is accessed using a key that does not exist within the current set of keys.

Solution:

1. **Scenario:** You have a dictionary `D = {'a': 1, 'b': 2}` and you attempt to access `D['c']`.
2. **Identify the Error:** Since `'c'` is not a key in the dictionary, Python cannot return a value.
3. **Select the Exception:** - `ValueError`: Used when a function gets an argument of the right type but inappropriate value (e.g., math domain errors). - `KeyError`: The specific exception for missing mapping keys. - `NameError`: Used when a variable name is not found in the local or global scope.
4. **Conclusion:** For dictionary-related lookup failures, `KeyError` is the standard exception.

Final Answer: The exception is `KeyError`.

Answer: (B)

Q43.

Solution

Concept: Resource management is a critical part of software engineering. Files are limited system resources. If a program opens too many files without closing them, it can lead to "resource exhaustion," causing the system to crash or refuse to open new files.

Solution:

1. **Standard Approach:** Normally, you must call `f.close()` manually. If an error occurs between `open()` and `close()`, the file may stay open.
2. **The `with` Statement:** This statement sets up a "Context Manager." Its primary benefit is that it ensures the file is **automatically closed** as soon as the block of code inside it finishes executing.
3. **Error Safety:** Even if an exception (like a `ZeroDivisionError`) occurs inside the `with` block, the context manager guarantees that the file's `__exit__` method is called, closing the file safely.
4. **Clean Code:** It makes the code more readable by clearly defining the scope in which the file is being used.
5. **Conclusion:** The main reason to use `with` is its automated and safe handling of file closures.

Final Answer: It automatically closes the file.

Answer: (B)

Q44.

Solution

Concept: The OSI (Open Systems Interconnection) model organizes networking into seven layers. The lowest layers (Physical and Data Link) deal with raw electrical signals and local hardware addressing, while the higher layers deal with routing and applications.

Solution:

1. **Analyze the Physical Layer (Layer 1):** This layer is concerned with the actual physical connection between devices. It defines the electrical, mechanical, and functional specifications for activating, maintaining, and deactivating the physical link.
2. **Identify the Unit of Data:** At this layer, data is not yet organized into frames or packets; it exists purely as a stream of **bits** (0s and 1s).
3. **Hardware examples:** Cables (Ethernet, Fiber), Hubs, and Repeaters all operate at the Physical Layer.
4. **Conclusion:** The Physical Layer is responsible for transmitting the raw bit stream over a physical medium.

Final Answer: The Physical Layer handles bit transmission.

Answer: (C)

Q45.

Solution

Concept: Application layer protocols are designed for specific tasks. For electronic mail, there are three primary protocols: SMTP (for sending), and POP3/IMAP (for receiving).

Solution:

1. **SMTP (Simple Mail Transfer Protocol):** As the name suggests, it is the standard protocol for **sending** email messages from one server to another or from a client to a server.
2. **FTP:** For file transfers.
3. **HTTP:** For web page transfers.
4. **SNMP (Simple Network Management Protocol):** Used for collecting and organizing information about managed devices on IP networks.
5. **Conclusion:** For email transmission, SMTP is the correct protocol.

Final Answer: The protocol is SMTP.

Answer: (B)

Q46.

Solution

Concept: Network security requires a combination of hardware and software solutions. A firewall acts as a barrier between a trusted internal network and untrusted external networks (like the internet).

Solution:

1. **Function of a Firewall:** It examines every packet of data entering or leaving the network. Based on a set of security rules, it either allows the packet to pass or ”drops” (blocks) it.
2. **Primary Goal:** To prevent unauthorized access to or from a private network. It can block specific ports, IP addresses, or types of traffic.
3. **Evaluate Options:** It doesn't physically speed up the internet, nor is its primary job to create backups (though it may protect the backup server).
4. **Conclusion:** Monitoring and controlling traffic is the defining role of a firewall.

Final Answer: Monitoring and controlling network traffic.

Answer: (B)

Q47.

Solution

Concept: Malware classification depends on how the software behaves. A Trojan Horse (named after the Greek myth) is a specific type of malware that relies on deception.

Solution:

1. **Analyze behavior:** A Trojan appears to be a legitimate, useful, or interesting program (like a free game, a calculator, or a system update).
2. **The ”Hidden” Payload:** Once the user installs and runs the program, it executes malicious code in the background—such as stealing passwords, deleting files, or creating a ”backdoor” for hackers.
3. **Key Distinction:** Unlike a virus or a worm, a Trojan does not self-replicate or infect other files. It depends on human trickery to spread.
4. **Conclusion:** Software that masks its malicious intent with a useful facade is a Trojan Horse.

Final Answer: It is a Trojan Horse.

Answer: (B)

Q48.

Solution

Concept: The Mesh Topology is the most robust but most expensive network design. It provides multiple redundant paths for data, ensuring that communication can continue even if one or more links fail.

Solution:

1. **Full Mesh:** In a "Full Mesh" topology, every single node is connected to every other node in the network.
2. **Dedicated Links:** These are point-to-point links that carry traffic only between the two nodes they connect.
3. **Reliability:** This topology provides high fault tolerance. If one cable is cut, the data is simply rerouted through another neighbor.
4. **Scalability Issues:** For n nodes, you need $n(n-1)/2$ cables. This makes it impractical for very large networks.
5. **Conclusion:** The description of a "dedicated point-to-point link to every other node" is the definition of a Mesh Topology.

Final Answer: The topology is Mesh.

Answer: (B)

Q49.

Solution

Concept: Interconnecting different types of networks requires specialized hardware that can translate between different protocol suites (e.g., connecting a TCP/IP network to an AppleTalk or IPX network).

Solution:

1. **Analyze Hub/Switch:** These operate at Layer 1 or 2 and connect devices within the *same* network using the same protocol.
2. **Analyze Bridge:** Connects two segments of the same type of network.
3. **Analyze Gateway:** A gateway is a protocol converter. It operates at all seven layers of the OSI model and acts as an "entry/exit point" for a network. Its primary job is to connect two networks that use completely different protocols.
4. **Conclusion:** For connecting networks with "different protocols," a Gateway is the necessary device.

Final Answer: The device is a Gateway.

Answer: (D)

Q50.

Solution

Concept: Secure communication on the web is achieved by layering standard protocols over security layers. The ”S” in HTTPS is the indicator that the connection is encrypted.

Solution:

1. **HTTPS Protocol:** Hypertext Transfer Protocol **Secure**.
2. **The Security Layer:** HTTPS uses SSL (Secure Sockets Layer) or, more commonly today, TLS (Transport Layer Security) to encrypt the data packets.
3. **Impact:** This prevents ”Man-in-the-Middle” attacks where a hacker could intercept and read your passwords or credit card numbers.
4. **Conclusion:** The ”S” stands for Secure.

Final Answer: It stands for Secure.

Answer: (C)

Answer Key

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