CUET-UG Computer Science Sample Paper - 20

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. What is the output of the following SQL query?

```
SELECT LENGTH(LTRIM('  CUET 2026  '));
```

(A) 9

(B) 11

(C) 14

(D) 12

Q2. Which SQL function would you use to find the day of the week (e.g., Monday, Tuesday) from the date '2026-05-07'?

(A) WEEKDAY()

(B) DAYNAME()

(C) DAYOFWEEK()

(D) DATE_NAME()

Q3. The result of `SELECT TRUNCATE(298.765, -2);` is:

(A) 298.76

(B) 298

(C) 200

(D) 300

Q4. Which string function is used to convert the first letter of each word to uppercase in a string?

(A) UPPER()

(B) INITCAP()

(C) CONVERT()

(D) Proper SQL does not have a single function for this; it requires a combination.

Q5. Evaluate: `SELECT INSTR('INTERNATIONAL', 'NA', 1, 2);`

(A) 6

(B) 11

(C) 5

(D) 10

Q6. To display the current time without the date, which of the following is correct?

(A) SELECT CURTIME();

(B) SELECT CURRENT_TIME();

(C) SELECT TIME(NOW());

(D) All of the above

Q7. What is the return type of the SQL `MONTH()` function?

(A) String

(B) Integer

(C) Date

(D) Float

Q8. The SQL statement `SELECT REPLACE('DATABASE', 'A', '@');` returns:

(A) D@T@B@SE

(B) D@TABASE

(C) D@T@BASE

(D) DATAB@SE

Q9. In Relational Algebra, which operation is used to combine all rows from two relations, excluding duplicates?

(A) Intersection ($\cap$)

(B) Cartesian Product ($\times$)

(C) Union ($\cup$)

(D) Set Difference ($-$)

Q10. Which of the following is a non-key attribute that depends on another non-key attribute, violating 3NF?

(A) Partial Dependency

(B) Transitive Dependency

(C) Functional Dependency

(D) Multi-valued Dependency

Q11. A _____ is a virtual table based on the result-set of an SQL statement.

(A) Metadata

(B) View

(C) Trigger

(D) Index

Q12. In a relationship where one record in Table A can be associated with multiple records in Table B, and vice versa, it is called:

(A) One-to-One

(B) One-to-Many

(C) Many-to-One

(D) Many-to-Many

Q13. Which protocol is responsible for resolving a known IP address into a MAC address?

(A) DNS

(B) DHCP

(C) ARP

(D) RARP

Q14. In which topology is every computer and network device connected to a single cable?

(A) Star

(B) Ring

(C) Bus

(D) Tree

Q15. The range of private IP addresses for Class C is:

(A) 10.0.0.0 to 10.255.255.255

(B) 172.16.0.0 to 172.31.255.255

(C) 192.168.0.0 to 192.168.255.255

(D) 169.254.0.0 to 169.254.255.255

Q16. A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as a _____.

(A) Stack

(B) Queue

(C) Tree

(D) Linked List

Q17. Convert the following prefix expression to infix: `+ × A B C`

(A) $A \times B + C$

(B) $(A \times B) + C$

(C) $A + (B \times C)$

(D) $A \times (B + C)$

Q18. The post-order traversal of a tree is:

(A) Root, Left, Right

(B) Left, Root, Right

(C) Left, Right, Root

(D) Right, Left, Root

Q19. Which of the following data structures is non-linear?

(A) Stack

(B) String

(C) Graph

(D) List

Q20. How many pointers are required to implement a Doubly Linked List node?

(A) 1

(B) 2

(C) 3

(D) 0

Q21. In a stack implemented using an array of size N, the `isFull()` condition is:

(A) top == 0

(B) top == N

(C) top == N - 1

(D) top == -1

Q22. Which operation in a stack has $O(n)$ complexity if the stack is implemented as a simple array and needs resizing?

(A) Push

(B) Pop

(C) Peek

(D) None

Q23. The process of accessing each element of a data structure exactly once is called _____.

(A) Sorting

(B) Searching

(C) Traversing

(D) Inserting

Q24. A Deque (Double Ended Queue) allows insertion and deletion at:

(A) Only the Front

(B) Only the Rear

(C) Both Front and Rear

(D) Only the Middle

Q25. If the sequence of push operations is 1, 2, 3 and there is one pop operation after each push, the sequence of popped elements is:

(A) 3, 2, 1

(B) 1, 2, 3

(C) 2, 1, 3

(D) 1, 3, 2

Q26. The condition `Rear == Front` in a circular queue typically signifies:

(A) Queue is full

(B) Queue is empty

(C) Queue has one element

(D) Queue is initialized but not used

Q27. Which sorting algorithm has a worst-case time complexity of $O(n \log n)$?

(A) Bubble Sort

(B) Selection Sort

(C) Merge Sort

(D) Insertion Sort

Q28. In Selection Sort, the number of swaps performed in the worst case is:

(A) $O(n^2)$

(B) $O(n \log n)$

(C) $O(n)$

(D) $O(1)$

Q29. Which searching algorithm is most efficient for an unsorted list of $1,000,000$ elements?

(A) Binary Search

(B) Linear Search

(C) Interpolation Search

(D) Hashing

Q30. What is the average number of comparisons in a Linear Search for a list of n elements?

(A) n

(B) $n/2$

(C) $\log n$

(D) n^2

Q31. The ”Divide and Conquer” strategy is used in which sorting algorithm?

(A) Insertion Sort

(B) Bubble Sort

(C) Quick Sort

(D) Selection Sort

Q32. Which sort is best suited for a list that is already almost sorted?

(A) Selection Sort

(B) Insertion Sort

(C) Quick Sort

(D) Heap Sort

Q33. In a Binary Search, if the target is smaller than the middle element, the next search range is:

(A) Lower Half

(B) Upper Half

(C) The whole list again

(D) The search terminates

Q34. Binary Search can be implemented using which programming technique?

(A) Iteration

(B) Recursion

(C) Both A and B

(D) Neither A nor B

Q35. The complexity of Bubble Sort in the best-case (with optimized flag) is:

(A) $O(n^2)$

(B) $O(n \log n)$

(C) $O(n)$

(D) $O(1)$

Q36. Which function is used to convert a Python object into a byte stream to be saved in a binary file?

(A) pickle.load()

(B) pickle.dump()

(C) file.write()

(D) file.save()

Q37. What is the output of `f.tell()` in Python file handling?

(A) Total size of the file

(B) Current position of the file pointer

(C) Name of the file

(D) The first line of the file

Q38. In Python, which keyword is used to manually trigger an exception?

(A) throw

(B) trigger

(C) raise

(D) except

Q39. To append data to the end of an existing text file, which mode is used?

(A) 'w'

(B) 'r+'

(C) 'a'

(D) 'w+'

Q40. What is the purpose of the `sys.argv` list in Python?

(A) To store environment variables

(B) To store command-line arguments

(C) To store system-level exceptions

(D) To store global variables

Q41. Which error occurs when you try to perform an operation on data of an inappropriate type (e.g., adding a string to an integer)?

(A) ValueError

(B) TypeError

(C) NameError

(D) SyntaxError

Q42. The `read(n)` method of a file object reads _____.

(A) n lines

(B) n bytes/characters

(C) The whole file except the last n bytes

(D) Only the n^{th} line

Q43. Which block handles any exception that is not specifically caught by previous `except` blocks?

(A) `except Exception:`

(B) `finally:`

(C) `else:`

(D) `try:`

Q44. Which switching technique is used by the Internet?

(A) Circuit Switching

(B) Packet Switching

(C) Message Switching

(D) Telephone Switching

Q45. Which malware type does not need a host file to replicate and spread?

(A) Virus

(B) Worm

(C) Trojan

(D) Logic Bomb

Q46. The protocol used to retrieve emails from a server and store them on a local device is:

(A) SMTP

(B) POP3

(C) HTTP

(D) FTP

Q47. What is 'Salting' in the context of network security?

(A) Adding random data to passwords before hashing

(B) Encrypting a hard drive

(C) Preventing SQL Injection

(D) A type of firewall configuration

Q48. Which layer of the OSI model is responsible for end-to-end communication and error recovery?

(A) Network Layer

(B) Transport Layer

(C) Session Layer

(D) Data Link Layer

Q49. Cookies are used in web communication to:

(A) Store user passwords permanently on the server

(B) Maintain state information across multiple requests

(C) Increase the speed of the CPU

(D) Prevent virus attacks

Q50. The process of transforming plain text into an unreadable format using a key is called:

(A) Decryption

(B) Encoding

(C) Encryption

(D) Masking

Detailed Solutions

Q1.

Solution

Concept:

The `LTRIM()` function in SQL is a string manipulation function used to remove all leading spaces (left-side spaces) from a given string. The `LENGTH()` function is used to calculate the total number of characters present in a string, including alphanumeric characters, special symbols, and trailing spaces (right-side spaces). In this specific query, the functions are nested, meaning the inner `LTRIM()` is executed first, and its result is passed to the outer `LENGTH()` function.

Solution:

1. The input string provided is `'  CUET 2026  '`. This string contains three leading spaces, the text ”CUET 2026”, and three trailing spaces.
2. First, the `LTRIM()` function is applied:

Original String: `'  CUET 2026  '`

After `LTRIM()`: `'CUET 2026  '`

The three spaces at the beginning are removed, but the three spaces at the end remain untouched.

3. Next, the `LENGTH()` function evaluates the result `'CUET 2026  '`.
4. Let us count the characters in `'CUET 2026  '`:

- ’C’, ’U’, ’E’, ’T’ (4 characters)
- Space between CUET and 2026 (1 character)
- ’2’, ’0’, ’2’, ’6’ (4 characters)
- Trailing spaces (3 characters)

5. Adding them together: 4 + 1 + 4 + 3 = 12.
6. Therefore, the total length of the string after trimming the left side is 12.

Final Answer: The output of the query is 12.

Answer: (D)

Q2.

Solution

Concept:

In SQL, date functions are used to extract specific parts of a date or to format a date into a readable string. To retrieve the name of the day (such as "Monday" or "Tuesday") corresponding to a specific date, different database systems provide specific built-in functions. While `WEEKDAY()` or `DAYOFWEEK()` return numeric values representing the day, `DAYNAME()` is specifically designed to return the full string representation of the day name.

Solution:

1. The requirement is to find the descriptive name of the day (e.g., Monday, Tuesday) for the date '2026-05-07'.

2. Let's analyze the given options:

- `WEEKDAY()`: This function returns the index of the day (e.g., 0 for Monday, 1 for Tuesday in some systems). It does not return the name.
- `DAYNAME()`: This function takes a date as an argument and returns the name of the day as a string (e.g., 'Thursday'). This matches the requirement perfectly.
- `DAYOFWEEK()`: This returns an index (e.g., 1 for Sunday, 2 for Monday). It is numeric rather than a string name.
- `DATE_NAME()`: This is not a standard function in MySQL, although `DATENAME()` exists in SQL Server with different syntax requirements.

3. For the date '2026-05-07', using `SELECT DAYNAME('2026-05-07');` would specifically return the string identifying that day on the calendar.

4. Thus, `DAYNAME()` is the correct function for this task.

Final Answer: The function used to find the day name is DAYNAME().

Answer: (B)

Q3.

Solution

Concept:

The `TRUNCATE(number, decimals)` function in SQL is used to shorten a number to a specified number of decimal places. Unlike the `ROUND()` function, `TRUNCATE()` does not consider the value of the next digit to decide whether to round up or down; it simply removes digits. If the `decimals` parameter is a positive integer, it affects the right side of the decimal point. If it is a negative integer, it affects the digits to the left of the decimal point (the integer part), replacing them with zeros.

Solution:

1. The query provided is `SELECT TRUNCATE(298.765, -2);`.
2. Here, the number is 298.765 and the decimal parameter is -2.
3. A negative value for the second parameter means we must truncate digits to the left of the decimal point. Specifically, -2 means we move two places to the left from the decimal.
4. Looking at 298.765:

- The first digit to the left is 8 (ones place).
- The second digit to the left is 9 (tens place).

5. Truncating at -2 means the digits in the tens and ones places (9 and 8) will be replaced by 0. The digits to the right of the decimal are also discarded.
6. The hundreds place digit (2) remains, and the rest become zero.
7. Result calculation:

- 298.765 $\rightarrow$ 200.

8. Note that if this were `ROUND(298.765, -2)`, the answer would be 300 because 98 is closer to 100. However, `TRUNCATE()` simply cuts off the value without rounding logic.

Final Answer: The result is 200.

Answer: (C)

Q4.

Solution

Concept:

String casing functions in SQL allow developers to modify the capitalization of text data. Common functions include `UPPER()` for full uppercase and `LOWER()` for full lowercase. However, the requirement to convert the first letter of each word to uppercase (Title Case or Proper Case) is a specific feature. While Oracle SQL provides the `INITCAP()` function, many other systems like MySQL do not have a single built-in function for this, often requiring a combination of `CONCAT`, `UPPER`, and `SUBSTR`.

Solution:

1. The goal is to identify which function converts the first letter of each word to uppercase.
2. Let's evaluate the options:

- `UPPER()`: This converts the entire string to capital letters. For example, 'sql' becomes 'SQL'. It does not perform word-specific casing.
- `INITCAP()`: This stands for "Initial Capital". It is a valid function in Oracle Database that does exactly what is described (e.g., 'database management' becomes 'Database Management').
- `CONVERT()`: This is generally used for changing data types or character set encodings, not for casing.
- `Option (D)`: "Proper SQL does not have a single function for this; it requires a combination." While true for MySQL/SQL Server, `INITCAP()` is the standard textbook answer for this concept.

3. Given the standard options in database examinations, `INITCAP()` is the intended answer for this specific functionality.

Final Answer: The function is INITCAP().

Answer: (B)

Q5.

Solution

Concept:

The `INSTR()` function is used to find the position of a substring within a string. The syntax is generally `INSTR(string, substring, [start_position], [occurrence])`.

- `string`: The main string to search.
- `substring`: The text to find.
- `start_position`: The position where the search begins (default is 1).
- `occurrence`: Which instance of the substring to find (default is 1).

Solution:

1. The query is `SELECT INSTR('INTERNATIONAL', 'NA', 1, 2);`.
2. Here:
- Main String = `'INTERNATIONAL'`
- Substring = `'NA'`
- Start Position = 1 (begin from the first character 'I')
- Occurrence = 2 (we need the position of the "second" time 'NA' appears).
3. Let's trace the string `'INTERNATIONAL'` index by index:
1:I, 2:N, 3:T, 4:E, 5:R, 6:N, 7:A, 8:T, 9:I, 10:O, 11:N, 12:A, 13:L.
4. Find the first 'NA':
It starts at index 6 (where 'N' is at 6 and 'A' is at 7).
5. Find the second 'NA':
Continuing from index 8, we find the next 'NA' starting at index 11 (where 'N' is at 11 and 'A' is at 12).
6. Since the query asks for the 2nd occurrence, the function returns the starting position of that instance.
7. The value returned is 11.

Final Answer: The evaluation result is 11.

Answer: (B)

Q6.

Solution

Concept:

In SQL, obtaining the current time of the system or database server is a common requirement for logging, timestamps, and scheduling. Different SQL dialects (like MySQL, PostgreSQL, and SQL Server) provide multiple aliases or functions to achieve this. These functions can return just the time component or a full timestamp (date and time), which can then be cast or filtered to show only the time portion.

Solution:

1. Let us examine the validity of each provided function:

- `CURTIME()`: This is a standard MySQL function that returns the current time in 'HH:MM:SS' format. It is a shorthand for `CURRENT_TIME()`.
- `CURRENT_TIME()`: This is a standard SQL function (supported by MySQL, PostgreSQL, etc.) that performs the same action as `CURTIME()`. It is part of the SQL standard.
- `TIME(NOW())`: The `NOW()` function returns the current date and time (e.g., '2026-05-07 11:40:30'). The `TIME()` function extracts only the time part from a datetime expression. Therefore, `TIME(NOW())` correctly results in the current time.

2. Since all three methods are syntactically correct and achieve the goal of displaying the current time without the date, the correct choice is "All of the above".

3. These functions are highly useful when you need to record the exact moment a transaction occurs without cluttering the output with date information.

Final Answer: All of the mentioned functions are correct.

Answer: (D)

Q7.

Solution

Concept:

The `MONTH()` function is a date-processing function used to extract the month part from a specific date or datetime expression. In programming and database management, it is crucial to understand whether a function returns a string (like 'January') or a numeric value (like 1) to ensure compatibility with arithmetic operations or comparisons in the `WHERE` clause.

Solution:

1. The `MONTH()` function takes a date as an input, such as '2026-05-07'.
2. When executed, the function identifies the month component of the date. In this case, the month is May.
3. Instead of returning the word 'May', the `MONTH()` function returns the numerical index of the month. For May, it returns the number 5.
4. The range of values returned by this function is 1 (for January) through 12 (for December).
5. In terms of data types, the number 5 is an integer (`INT`).
6. If a user required the string 'May', they would use a different function like `MONTHNAME()`. Since `MONTH()` specifically returns the numeric representation, its return type is Integer.

Final Answer: The return type of the MONTH() function is Integer.

Answer: (B)

Q8.

Solution

Concept:

The `REPLACE()` function is a string function used to replace all occurrences of a specified substring within a string with a new substring. The syntax is `REPLACE(string, from_string, to_string)`. It is important to note that this function is case-sensitive and, most importantly, it replaces "every" instance of the target character or word found in the source string, not just the first one.

Solution:

1. We are given the statement: `SELECT REPLACE('DATABASE', 'A', '@');`.
2. Here:
- The original string is `'DATABASE'`.
- The character to be replaced is `'A'`.
- The replacement character is `'@'`.
3. Let's look at the string `'DATABASE'` and identify all occurrences of the letter `'A'`:
- D - **A** (1st occurrence)
- T - **A** (2nd occurrence)
- B - **A** (3rd occurrence)
- S - E
4. There are three `'A'` characters in total.
5. The `REPLACE()` function will substitute every `'A'` with `'@'`.
6. Performing the substitution:
- D + @ + T + @ + B + @ + S + E = `'D@T@B@SE'`.
7. Comparing this to the options, we see that option (A) matches this result exactly.

Final Answer: The resulting string is D@T@B@SE.

Answer: (A)

Q9.

Solution

Concept:

Relational Algebra is a procedural query language used to model the data stored in relations and define operations on them. Set operations are a core part of this logic. The Union operation ($\cup$) is used to combine the tuples (rows) from two compatible relations. A key property of the Union operation in relational algebra is that it automatically eliminates duplicate rows, ensuring the resulting set follows mathematical set theory where each element is unique.

Solution:

1. The question asks for an operation that combines all rows from two relations while excluding duplicates.

2. Let's evaluate the set operations:

- **Union** ($\cup$): This operation returns a relation containing all tuples that are in relation A, or in relation B, or in both. Duplicates are removed by definition.

- **Intersection** ($\cap$): This returns only the tuples that are present in "both" relations. It does not combine all rows.

- **Cartesian Product** ($\times$): This combines every row of the first relation with every row of the second relation, resulting in $m \times n$ rows. It does not handle "union" logic or duplicate removal in this context.

- **Set Difference** ($-$): This returns tuples present in the first relation but not the second.

3. Therefore, the Union operation is the correct mathematical and logical choice for combining sets and removing duplicates.

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

Answer: (C)

Q10.

Solution

Concept:

Normalization is the process of organizing data in a database to reduce redundancy. The Third Normal Form (3NF) has a specific requirement: a table must already be in 2NF, and all its non-key attributes must be functionally dependent ”only” on the primary key. If a non-key attribute depends on another non-key attribute, this is known as a Transitive Dependency. Eliminating such dependencies is the primary goal of achieving 3NF.

Solution:

1. We need to identify the dependency where a non-key attribute depends on another non-key attribute.

2. Definitions of dependencies:

- **Partial Dependency**: Occurs when a non-key attribute depends on only ”part” of a composite primary key. This violates 2NF.
- **Functional Dependency**: A general relationship where the value of one attribute determines the value of another.
- **Transitive Dependency**: This occurs when attribute A determines B, and B determines C (where A is the primary key and B, C are non-key attributes). Here, C depends on B, which is a non-key attribute. This is the definition provided in the question.
- **Multi-valued Dependency**: Occurs when one or more rows in a table imply the presence of one or more other rows. This is related to 4NF.

3. Since the dependency between two non-key attributes is specifically called a transitive dependency, it is the correct answer.

Final Answer: The dependency is Transitive Dependency.

Answer: (B)

Q11.

Solution

Concept:

In database management systems, a View is a logical representation of data. Unlike a regular table, a View does not store data physically in the database; it is essentially a stored SQL query. When you access a View, the database engine executes the underlying query and presents the result as if it were a real table. This is why it is often called a ”virtual table.” Views are used to simplify complex queries, provide security by restricting access to specific columns, and maintain a consistent data interface.

Solution:

1. Let’s analyze the terms provided in the options:

- **Metadata**: This refers to ”data about data,” describing the structure, constraints, and properties of the database objects. It is not a table.
- **View**: This is a database object that represents the result-set of a predefined SQL `SELECT` statement. It behaves like a table but is created dynamically.
- **Trigger**: This is a procedural code that automatically executes in response to certain events (like `INSERT`, `UPDATE`, or `DELETE`) on a particular table.
- **Index**: This is a data structure used to speed up the retrieval of rows from a table.

2. The definition ”virtual table based on the result-set of an SQL statement” is the formal definition of a View.

3. For example, if we have a complex join query that we use frequently, we can save it as a View and simply run `SELECT * FROM view_name;` instead of writing the whole join again.

Final Answer: The virtual table is known as a View.

Answer: (B)

Q12.

Solution

Concept:

In database design, relationships define how data in one table is connected to data in another. These are visualized through Entity-Relationship (ER) diagrams and implemented using Foreign Keys. A relationship is categorized based on ”cardinality,” which describes how many instances of one entity can be associated with how many instances of another entity.

Solution:

1. We are given a specific scenario: ”one record in Table A can be associated with multiple records in Table B, and vice versa.”

2. Let’s break this down:

- From A to B: One A → Many B.
- From B to A: One B → Many A.

3. When both directions allow for multiple associations, the resulting relationship is called a Many-to-Many (M:N) relationship.

4. Example: A ”Student” can enroll in many ”Courses,” and a single ”Course” can have many ”Students.” This is a classic Many-to-Many scenario.

5. In physical database design, Many-to-Many relationships cannot be implemented directly between two tables. They require a third ”junction table” or ”associative table” to store the pairs of related IDs.

Final Answer: The relationship is Many-to-Many.

Answer: (D)

Q13.

Solution

Concept:

In a local area network (LAN), communication between devices happens at the Data Link Layer (Layer 2) using MAC addresses. However, applications and users typically use IP addresses (Layer 3). To send data to a device, the sending system must know the physical MAC address of the destination that corresponds to its logical IP address. The protocol that bridges this gap is the Address Resolution Protocol (ARP).

Solution:

1. The goal is to resolve a known IP address into a hardware MAC address.

2. Let's check the protocols:

- **DNS (Domain Name System)**: Resolves domain names (like google.com) into IP addresses.
- **DHCP (Dynamic Host Configuration Protocol)**: Automatically assigns IP addresses to devices on a network.
- **ARP (Address Resolution Protocol)**: Used by a host to find the MAC address of another host on the same network given its IP address. This matches the question perfectly.
- **RARP (Reverse ARP)**: The opposite of ARP; it allows a device to find its own IP address if it only knows its MAC address (mostly obsolete now, replaced by DHCP).

3. When a computer needs to communicate, it broadcasts an ARP Request: "Who has IP X.X.X.X? Tell me your MAC." The target replies with an ARP Reply containing its MAC address.

Final Answer: The protocol is ARP.

Answer: (C)

Q14.

Solution

Concept:

Network topology describes the physical or logical arrangement of nodes (computers, printers, etc.) and the connections between them. Different topologies have different methods of data transmission and different vulnerabilities. A topology where all devices share a single backbone or communication line is one of the simplest forms of networking.

Solution:

1. The description given is: ”every computer and network device connected to a single cable.”
2. Let’s analyze the topologies:

- **Star**: All devices are connected to a central hub or switch. If a device fails, others are fine. If the hub fails, the network goes down.
- **Ring**: Each device is connected to exactly two other devices, forming a circular path for signals.
- **Bus**: All nodes are connected to a single central cable, called the ”bus” or ”backbone.” Data travels along the cable and is received by the intended recipient. If the main cable breaks, the entire network fails. This matches the question description.
- **Tree**: A hierarchical topology combining characteristics of bus and star topologies.

3. The Bus topology was commonly used in early Ethernet networks (using coaxial cables) and fits the definition of sharing a single cable perfectly.

Final Answer: This topology is known as the Bus topology.

Answer: (C)

Q15.

Solution

Concept:

To prevent the exhaustion of public IP addresses and to provide security, certain ranges of IP addresses are reserved for private use within internal networks (like homes, schools, or offices). These addresses are defined by RFC 1918. They are not routable on the public internet. IP addresses are divided into classes (A, B, and C), and each class has a dedicated private range.

Solution:

1. We need to identify the Class C private IP range.
2. Let's look at the standard reserved private ranges:
- **Class A**: 10.0.0.0 to 10.255.255.255. This provides a single huge network with millions of addresses.
- **Class B**: 172.16.0.0 to 172.31.255.255. This provides 16 contiguous networks.
- **Class C**: 192.168.0.0 to 192.168.255.255. This is the most common range used in home routers and small office networks. It provides 256 networks, each supporting up to 254 hosts.
3. Analyzing the options:
- Option (A) is Class A.
- Option (B) is Class B.
- Option (C) is 192.168.0.0 to 192.168.255.255, which is Class C.
- Option (D) is the APIPA (Automatic Private IP Addressing) range, used when a DHCP server is unavailable.
4. Therefore, Option (C) is the correct Class C private range.

Final Answer: The Class C private range is 192.168.0.0 to 192.168.255.255.

Answer: (C)

Q16.

Solution

Concept:

A data structure is a specialized format for organizing, processing, retrieving, and storing data. Linear data structures are those where elements are arranged in a sequence. Among these, the Queue is a fundamental structure that follows the First-In, First-Out (FIFO) principle. This means the first element added to the queue will be the first one to be removed. In a Queue, two distinct ends are used: one for adding elements and another for removing them.

Solution:

1. We are looking for a data structure where:

- Deletion occurs at one end called the ”front.”
- Insertion occurs at the other end called the ”rear.”

2. Let’s compare the options:

- **Stack**: Follows Last-In, First-Out (LIFO). Both insertion (push) and deletion (pop) happen at the same end (top).
- **Queue**: Follows First-In, First-Out (FIFO). Insertion happens at the rear and deletion happens at the front. This matches the definition in the question.
- **Tree**: A non-linear, hierarchical data structure.
- **Linked List**: A linear structure, but insertion/deletion can technically happen anywhere (start, end, or middle) depending on the pointer logic.

3. The specific restriction of insertion at one end and deletion at the other is the defining characteristic of a Queue. This is similar to a real-life queue at a ticket counter.

Final Answer: The data structure described is a Queue.

Answer: (B)

Q17.

Solution

Concept:

Mathematical expressions can be written in three notations: Infix (operator between operands, like $A + B$), Prefix (operator before operands, like $+AB$), and Postfix (operator after operands, like $AB+$). To convert from Prefix to Infix, we evaluate the expression from right to left. When an operator is encountered, it is applied to the operands that follow it.

Solution:

1. The given prefix expression is: `+ * A B C`.
2. To convert to Infix, we scan from right to left:

- Scan 'C': It is an operand. Push to a stack.
- Scan 'B': It is an operand. Push to the stack.
- Scan 'A': It is an operand. Push to the stack.
- Scan '*': It is an operator. Pop the last two operands ('A' and 'B') and place the operator between them: `(A * B)`. Now push this back to the stack.
- Scan '+': It is an operator. Pop the current top elements: `(A * B)` and 'C'. Place the operator between them: `(A * B) + C`.

3. Following the order of operations, the multiplication is performed first, and then the result is added to C.
4. Thus, the infix equivalent is `(A * B) + C`.

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

Answer: (B)

Q18.

Solution

Concept:

Tree traversal refers to the process of visiting each node in a tree exactly once. For binary trees, there are three standard depth-first traversals:

- **In-order**: Left subtree, Root, Right subtree.
- **Pre-order**: Root, Left subtree, Right subtree.
- **Post-order**: Left subtree, Right subtree, Root.

The name of the traversal (Pre, In, Post) refers to the position of the ”Root” node in the sequence.

Solution:

1. The question specifically asks for the order of Post-order traversal.
2. In Post-order traversal, the algorithm recursively visits the left child and the right child before visiting the current node (the root of that specific subtree).
3. Let’s look at the logic:
- Step 1: Traverse the left subtree.
- Step 2: Traverse the right subtree.
- Step 3: Visit the root node.
4. This results in the sequence: Left, Right, Root.
5. Post-order is commonly used in applications like deleting a tree or evaluating mathematical expressions in postfix notation.

Final Answer: The post-order traversal is Left, Right, Root.

Answer: (C)

Q19.

Solution

Concept:

Data structures are broadly classified into two categories:

- **Linear Data Structures**: Elements are arranged in a linear sequence where each element (except the first and last) has a unique predecessor and successor. Examples include Arrays, Stacks, Queues, and Linked Lists.
- **Non-linear Data Structures**: Elements are not arranged in a sequence. One element can be connected to multiple other elements, forming a hierarchical or interconnected relationship. Examples include Trees and Graphs.

Solution:

1. We need to identify the non-linear data structure from the list.
2. Analyzing the options:
- **Stack**: A linear structure where elements are added and removed in a specific order (LIFO).
- **String**: A linear sequence of characters.
- **Graph**: A collection of nodes (vertices) and edges that connect them. A node can have connections to many other nodes, forming a web-like structure. This is a non-linear data structure.
- **List**: A general term for a linear sequence of elements.
3. Since Graphs and Trees are the primary examples of non-linear data structures, Graph is the correct choice here.

Final Answer: The non-linear data structure is Graph.

Answer: (C)

Q20.

Solution

Concept:

A Linked List consists of nodes, where each node contains data and references (pointers) to other nodes. In a Singly Linked List, each node has one pointer to the next node. In a Doubly Linked List (DLL), the structure is enhanced to allow traversal in both directions: forward and backward. To achieve this, each node must maintain additional metadata about its neighbors.

Solution:

1. Let's define the structure of a node in a Doubly Linked List.
2. Every node in a DLL contains three parts:

- **Data**: The actual value stored in the node.
- **Next Pointer**: A reference to the succeeding node in the list.
- **Prev (Previous) Pointer**: A reference to the preceding node in the list.

3. Since the node needs to point to the element before it and the element after it, it requires exactly two pointers.
4. These two pointers enable operations like deleting a node without needing to traverse from the head to find the predecessor, which is a major advantage over singly linked lists.

Final Answer: Two pointers are required to implement a Doubly Linked List node.

Answer: (B)

Q21.

Solution

Concept:

A stack is a linear data structure that follows the Last-In, First-Out (LIFO) principle. When a stack is implemented using a fixed-size array, it has a capacity limit defined by the size of the array (N). A pointer or index variable, typically called `top`, is used to keep track of the most recently added element. The value of `top` changes as elements are pushed or popped. If the array is indexed from 0 to $N - 1$, the stack is considered full when the `top` index reaches the last available slot in the array.

Solution:

1. Let the size of the array be N.
2. In a 0-indexed array, the first element is at index 0 and the last element is at index $N - 1$.
3. When the stack is empty, `top` is usually initialized to -1.
4. When the first element is added, `top` becomes 0.
5. When the second element is added, `top` becomes 1.
6. Following this logic, when the N^{th} element (the last possible element) is added, the `top` pointer will point to the index $N - 1$.
7. At this stage, any further attempt to push an element will result in a "Stack Overflow" error.
8. Therefore, the condition to check if the stack is full is `top == N - 1`.

Final Answer: The isFull() condition is top == N - 1.

Answer: (C)

Q22.

Solution

Concept:

Big O notation is used to describe the time complexity or performance of an algorithm. For a stack implemented with a simple array, most operations like `Push`, `Pop`, and `Peek` take constant time, $O(1)$, because they only involve accessing or modifying the element at the `top` index. However, if the array has a fixed size and becomes full, it cannot accept new elements unless it is resized. Dynamic resizing involves creating a new, larger array and copying all existing elements into it.

Solution:

1. In a standard array-based stack, `Push` is $O(1)$ because we simply increment `top` and insert the value.
2. `Pop` and `Peek` are also $O(1)$ as they only access the current `top` index.
3. If the stack is implemented using a dynamic array (like Python's list or Java's ArrayList) and it reaches its current capacity n, a `Push` operation triggers a resize.
4. During resizing:

- A new array of size (typically) $2n$ is allocated.
- All n elements from the old array are copied to the new array one by one.

5. This copying process requires n operations, making the time complexity $O(n)$ for that specific push.
6. While the "amortized" time complexity remains $O(1)$, the worst-case complexity for a single push requiring resizing is $O(n)$.

Final Answer: The Push operation has $O(n)$ complexity during resizing.

Answer: (A)

Q23.

Solution

Concept:

Data structure operations are the various actions that can be performed on the data stored within them. Common operations include insertion, deletion, searching, sorting, and merging. One fundamental operation is systematic visitation. In many algorithms, it is necessary to go through the data structure to process every single element, perhaps to print them, count them, or perform a specific calculation on each.

Solution:

1. The question describes the process of "accessing each element of a data structure exactly once."
2. Let's define the options:

- **Sorting**: Arranging elements in a particular order (ascending or descending).
- **Searching**: Finding the location of a specific target element.
- **Traversing**: The act of moving through the data structure so that each element is visited exactly once. This matches the definition perfectly.
- **Inserting**: Adding a new element into the data structure.

3. Traversal is a prerequisite for many other operations. For instance, to search for an element in an unsorted list, one must traverse the list. To print the elements of a linked list, one must perform a traversal starting from the head node.

Final Answer: The process is called Traversing.

Answer: (C)

Q24.

Solution

Concept:

A Deque, or Double-Ended Queue, is a more generalized form of a linear data structure than a standard Queue or Stack. While a regular Queue follows the FIFO principle (insertion at rear, deletion at front) and a Stack follows the LIFO principle (insertion and deletion at the top), a Deque removes these directional restrictions. It provides the flexibility to behave as either a stack or a queue depending on which ends are used for operations.

Solution:

1. A Deque is characterized by its ability to perform operations at both of its ends.
2. Specifically:

- Elements can be added to the **Front**.
- Elements can be removed from the **Front**.
- Elements can be added to the **Rear**.
- Elements can be removed from the **Rear**.

3. Because of this dual-ended access, a Deque is often used in complex algorithms like finding the maximum in a sliding window or maintaining a browser's history where you can go both forward and backward.
4. Options (A) and (B) describe restricted structures (like a stack or a standard queue), while Option (C) accurately captures the "Double-Ended" nature of the Deque.

Final Answer: A Deque allows insertion and deletion at both Front and Rear.

Answer: (C)

Q25.

Solution

Concept:

A stack follows the LIFO (Last-In, First-Out) principle. This means the most recent element pushed onto the stack is the first one to be removed by a pop operation. To determine the sequence of popped elements, one must carefully track the ”state” of the stack after every individual push and pop operation.

Solution:

1. The sequence of operations is: ”one pop operation after each push.”
2. Let’s trace the steps:
- **Step 1**: Push 1. (Stack: [1]).
- **Step 2**: Pop. The element popped is **1**. (Stack: []).
- **Step 3**: Push 2. (Stack: [2]).
- **Step 4**: Pop. The element popped is **2**. (Stack: []).
- **Step 5**: Push 3. (Stack: [3]).
- **Step 6**: Pop. The element popped is **3**. (Stack: []).
3. The sequence of elements as they were popped is 1, 2, 3.
4. Contrast this with a scenario where all pushes happen first (1, 2, 3) followed by all pops; in that case, the popped sequence would be 3, 2, 1.
5. Because each element was pushed and immediately popped before the next element arrived, the output order matches the input order.

Final Answer: The sequence of popped elements is 1, 2, 3.

Answer: (B)

Q26.

Solution

Concept:

A Circular Queue is a linear data structure that overcomes the limitation of a standard queue where space is wasted because the `Rear` pointer cannot wrap around to the beginning once it reaches the end of the array. In a circular queue, the last position is connected back to the first position. Two pointers, `Front` and `Rear`, are used. However, the condition `Rear == Front` can be ambiguous because it is used differently across implementations—sometimes to indicate an empty queue and sometimes to indicate that the queue has exactly one element remaining.

Solution:

1. In most standard textbook implementations, a circular queue is initialized with `Front = -1` and `Rear = -1`.
2. When the first element is added, both `Front` and `Rear` are set to 0. At this point, `Rear == Front` is true, and the queue has one element.
3. As more elements are added, `Rear` moves forward. As elements are removed, `Front` moves forward.
4. If we continue to remove elements until only one remains, `Front` and `Rear` will eventually point to the same index.
5. After that last element is popped, the pointers are typically reset to -1, or `Front` is moved past `Rear`.
6. Therefore, the state `Rear == Front` (when they are not at their null/initial state) indicates that there is exactly one element left in the queue.

Final Answer: The condition signifies that the queue has one element.

Answer: (C)

Q27.

Solution

Concept:

Sorting algorithms are evaluated based on their time complexity in different scenarios (Best, Average, and Worst case). Common algorithms like Bubble Sort, Selection Sort, and Insertion Sort have a worst-case complexity of $O(n^2)$, making them inefficient for large datasets. In contrast, advanced algorithms like Merge Sort use a divide-and-conquer strategy to split the data into smaller chunks, resulting in a much more efficient worst-case time complexity.

Solution:

1. Let's review the worst-case complexities of the given sorting algorithms:

- **Bubble Sort**: $O(n^2)$ because every element is compared with every other element in nested loops.
- **Selection Sort**: $O(n^2)$ as it repeatedly finds the minimum element from the unsorted part.
- **Insertion Sort**: $O(n^2)$ when the data is sorted in reverse order.
- **Merge Sort**: $O(n \log n)$. This algorithm consistently splits the array in half ($\log n$ levels) and performs a linear merge (n operations) at each level.

2. Unlike Quick Sort, which can degrade to $O(n^2)$ in its worst case if the pivot is poor, Merge Sort is guaranteed to perform at $O(n \log n)$ regardless of the initial order of the data.

3. This makes Merge Sort highly reliable for predictable performance in critical systems.

Final Answer: Merge Sort has a worst-case complexity of $O(n \log n)$.

Answer: (C)

Q28.

Solution

Concept:

Selection Sort works by repeatedly finding the minimum element from the unsorted portion of the array and swapping it with the first element of that portion. While the number of "comparisons" in Selection Sort is always high ($O(n^2)$), one of its unique characteristics is how it handles "swaps." Unlike Bubble Sort, which might perform many swaps in a single pass, Selection Sort performs at most one swap per outer loop iteration.

Solution:

1. In Selection Sort, there are two nested loops. The inner loop finds the minimum value, and the outer loop places it in the correct position.
2. For an array of size n, the outer loop runs $n - 1$ times.
3. In each iteration of the outer loop, exactly one swap is performed to put the identified minimum element into its sorted position.
4. Even in the worst-case scenario (where every element is in the wrong place), the algorithm only performs a total of $n - 1$ swaps.
5. In Big O notation, $n - 1$ is represented as $O(n)$.
6. This makes Selection Sort very efficient in terms of memory writes (swaps) compared to other $O(n^2)$ algorithms like Bubble Sort, though its total time complexity remains $O(n^2)$.

Final Answer: The number of swaps in the worst case is O(n).

Answer: (C)

Q29.

Solution

Concept:

Searching efficiency depends heavily on whether the data is sorted or organized in a specific structure. Linear Search scans every element, while Binary Search requires the list to be sorted. However, when dealing with a massive dataset (like 1,000,000 elements) that is currently "unsorted," sorting it first to perform a single search is often more expensive than just scanning it. If frequent searches are needed, Hashing is the most efficient approach as it provides near-constant time access.

Solution:

1. Let's analyze the options for an "unsorted" list:

- **Binary Search**: Requires the list to be sorted. Sorting 10^6 elements takes $O(n \log n)$ time, which is too much for just one search.
- **Linear Search**: Takes $O(n)$ time. For $1,000,000$ elements, it takes at most $1,000,000$ comparisons.
- **Interpolation Search**: Also requires sorted data and a specific distribution.
- **Hashing**: This involves mapping data to a unique index using a hash function. If the data is stored in a Hash Table, the search time is $O(1)$ on average.

2. While Linear Search is the only basic algorithm that works on "any" unsorted list, Hashing is technically the "most efficient" way to organize and search through a large volume of data if we have the choice of data structure.

3. In a standard computational context for "efficiency," Hashing is the superior method for large-scale retrieval.

Final Answer: Hashing is the most efficient for such a large volume of data.

Answer: (D)

Q30.

Solution

Concept:

Linear Search (or Sequential Search) involves checking each element of a list one by one until a match is found or the end of the list is reached. To find the ”average” number of comparisons, we consider all possible positions where the target element could be found and calculate the mean value of the work done.

Solution:

1. Suppose we have a list of n elements.
2. If the element is at the 1st position, we make 1 comparison.
3. If the element is at the 2nd position, we make 2 comparisons.
4. If the element is at the n^{th} position, we make n comparisons.
5. Assuming the element is equally likely to be in any position, the average number of comparisons is the sum of all comparisons divided by n:

$$\text{Average} = \frac{1 + 2 + 3 + \cdots + n}{n}$$

6. Using the formula for the sum of the first n integers: $\frac{n(n+1)}{2}$.
7. Dividing by n:

$$\text{Average} = \frac{n(n+1)}{2n} = \frac{n+1}{2}$$

8. For large values of n, $\frac{n+1}{2}$ is approximately $n/2$.
9. This means on average, we have to look through half the list to find what we are searching for.

Final Answer: The average number of comparisons is n/2.

Answer: (B)

Q31.

Solution

Concept:

Divide and Conquer is an algorithmic paradigm that solves a problem by breaking it down into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem. This technique is highly efficient for sorting and searching. In sorting, it typically involves partitioning an array around a pivot or splitting it into halves recursively.

Solution:

1. Let's analyze the strategies of the given sorting algorithms:

- **Insertion Sort**: Uses an incremental approach, placing one element at a time in its correct position.
- **Bubble Sort**: Uses a comparison-based approach, repeatedly swapping adjacent elements.
- **Quick Sort**: Uses Divide and Conquer. It picks an element as a 'pivot' and partitions the given array around the picked pivot. It then recursively sorts the sub-arrays to the left and right of the pivot.
- **Selection Sort**: Uses a greedy approach, repeatedly finding the minimum element and placing it at the beginning.

2. Quick Sort and Merge Sort are the two most prominent examples of the Divide and Conquer strategy in sorting.

3. In Quick Sort, the "Divide" step is the partitioning, and the "Conquer" step is the recursive sorting of sub-arrays.

Final Answer: The Divide and Conquer strategy is used in Quick Sort.

Answer: (C)

Q32.

Solution

Concept:

The performance of sorting algorithms can vary significantly based on the initial arrangement of the input data. While $O(n \log n)$ algorithms are generally faster for large random datasets, certain $O(n^2)$ algorithms perform exceptionally well when the list is "nearly sorted." A list is nearly sorted if each element is at most a few positions away from its final sorted position. In such cases, an algorithm that minimizes comparisons and swaps for already ordered elements is ideal.

Solution:

1. Let's look at how algorithms behave with nearly sorted data:

- **Selection Sort**: Always performs $O(n^2)$ comparisons regardless of data order. It is not adaptive.
- **Insertion Sort**: It is highly adaptive. If the list is already sorted, it only performs $n - 1$ comparisons and 0 swaps, resulting in $O(n)$ time complexity. For nearly sorted data, it remains very close to $O(n)$.
- **Quick Sort**: Can actually perform poorly (approaching $O(n^2)$) on sorted data if the pivot selection is not optimized (like choosing the first or last element).
- **Heap Sort**: Consistently $O(n \log n)$, but has higher overhead than Insertion Sort for small or nearly sorted lists.

2. Insertion Sort is the preferred choice for nearly sorted lists or for small datasets because of its simplicity and efficiency in these specific scenarios.

Final Answer: Insertion Sort is best suited for almost sorted lists.

Answer: (B)

Q33.

Solution

Concept:

Binary Search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one. This "halving" logic relies on the fact that the data is ordered, allowing the algorithm to discard half of the remaining elements in a single step.

Solution:

1. In Binary Search, we first compare the target value with the middle element of the sorted array.
2. If the target is exactly equal to the middle element, the search is successful.
3. If the target is **smaller** than the middle element:
- Because the array is sorted in ascending order, all elements to the right of the middle are also larger than the target.
- Therefore, the target must be located in the left portion of the current range.
- This range is known as the **Lower Half**.
4. If the target were larger, we would search the Upper Half.
5. By discarding the Upper Half, we effectively reduce the problem size by 50% in one comparison.

Final Answer: The next search range is the Lower Half.

Answer: (A)

Q34.

Solution

Concept:

Algorithms can often be implemented using different programming paradigms. **Iteration** uses loops (like `while` or `for`) to repeat a set of instructions. **Recursion** is a technique where a function calls itself to solve smaller instances of the same problem. Binary Search is a classic example that fits both paradigms perfectly because its logic involves repeating the same "halving" process on progressively smaller ranges.

Solution:

1. **Iterative Implementation**: Uses a `while` loop that continues as long as the `low` index is less than or equal to the `high` index. It updates these pointers in each step.
2. **Recursive Implementation**: A function takes the `low` and `high` indices as parameters. If the target isn't found at the midpoint, the function calls itself with a new `low` or `high` value.
3. Both methods are valid:
- Iteration is often more memory-efficient as it avoids the overhead of the call stack.
- Recursion is often considered more elegant and easier to read.
4. Therefore, Binary Search can be implemented using both techniques.

Final Answer: Both Iteration and Recursion can be used.

Answer: (C)

Q35.

Solution

Concept:

Bubble Sort works by repeatedly stepping through the list, comparing adjacent elements and swapping them if they are in the wrong order. In its standard form, it always takes $O(n^2)$ time. However, it can be optimized by using a flag (boolean variable) to check if any swaps occurred during a pass. If no swaps occurred, it means the list is already sorted, and the algorithm can terminate early.

Solution:

1. In the best-case scenario for Bubble Sort, the input list is already sorted (e.g., $[1, 2, 3, 4, 5]$).
2. Without optimization, the algorithm would still run its nested loops n^2 times.
3. With an **optimized flag**:

- The outer loop starts its first pass.
- The inner loop compares all adjacent pairs: $(1, 2), (2, 3), (3, 4), (4, 5)$.
- Since no elements are out of order, no swaps are made.
- The flag remains `false`.
- The algorithm checks the flag, sees no swaps were made, and breaks the outer loop immediately.

4. Total work done: One pass through the n elements.
5. This results in a time complexity of $O(n)$.

Final Answer: The best-case complexity of optimized Bubble Sort is O(n).

Answer: (C)

Q36.

Solution

Concept:

Python's `pickle` module is used for serializing and de-serializing Python object structures. Serialization, also known as "pickling," is the process whereby a Python object hierarchy (like a dictionary, list, or class instance) is converted into a byte stream. This byte stream can then be stored in a binary file or transmitted over a network. The reverse process is called "unpickling."

Solution:

1. To save a Python object into a binary file, we use the `pickle` module.
2. Let's look at the specific functions provided by this module:

- `pickle.dump(obj, file)`: This function is used to write the serialized byte stream of the object `obj` into the opened file object `file`. This is exactly what the question asks for.
- `pickle.load(file)`: This function reads a byte stream from a binary file and reconstructs it back into a Python object.

3. Functions like `file.write()` are used for writing strings or bytes directly to a file but do not handle the complex logic of converting Python objects to byte streams automatically.
4. Therefore, `pickle.dump()` is the correct function for saving an object as a byte stream.

Final Answer: The function used is pickle.dump().

Answer: (B)

Q37.

Solution

Concept:

When a file is opened in Python, a ”file pointer” (or cursor) is used to keep track of the current position within the file. All read and write operations occur relative to this pointer. As you read or write data, the pointer moves forward. Python provides two main methods to manage this pointer: `tell()` to report the location and `seek()` to change the location.

Solution:

1. The `f.tell()` method is a built-in file object method.
2. It takes no arguments and returns an integer.
3. This integer represents the current position of the file pointer in the file, measured in bytes from the beginning of the file.
4. For example, if you open a file and read 10 characters, `f.tell()` will return 10.
5. This is different from:

- The total size of the file (which requires `os.path.getsize()` or seeking to the end).
- The name of the file (which is accessed via `f.name`).
- The contents of the file.

6. Thus, `f.tell()` is specifically for tracking the pointer's position.

Final Answer: The output is the current position of the file pointer.

Answer: (B)

Q38.

Solution

Concept:

Exception handling is a mechanism to handle runtime errors so that the normal flow of the application can be maintained. While Python automatically triggers exceptions for errors like division by zero or missing files, developers often need to ”force” an exception to occur when specific business logic is violated. This is known as ”raising” or ”throwing” an exception.

Solution:

1. Let's examine the keywords used for exceptions in various languages and Python:

- **throw**: Used in languages like C++, Java, and JavaScript to trigger an exception. It is not a keyword in Python.
- **trigger**: Not a standard keyword in most programming languages for exception handling.
- **raise**: This is the specific Python keyword used to signal that an exceptional condition has occurred. For example: `raise ValueError("Invalid input")`.
- **except**: This keyword is used to ”catch” or handle an exception that has already been raised; it does not trigger one.

2. Therefore, in Python, the `raise` statement is the correct way to manually trigger an exception.

Final Answer: The keyword used is raise.

Answer: (C)

Q39.

Solution

Concept:

When opening a file in Python using the `open()` function, a ”mode” must be specified. The mode determines what operations (read, write, append) are allowed and how the file pointer is positioned. Choosing the wrong mode can lead to accidental data loss, such as overwriting an existing file.

Solution:

1. Let’s analyze the common file modes:
- `'w'` (Write): Opens a file for writing. If the file exists, it **truncates (deletes)** the content. If it doesn’t exist, it creates it.
- `'r+'` (Read/Write): Opens the file for both reading and writing. The pointer is placed at the start. It does not delete existing content but will overwrite character by character from the start.
- `'a'` (Append): Opens the file for appending. The file pointer is automatically placed at the **end** of the file. Existing data is preserved, and new data is added after it.
- `'w+'` (Write/Read): Similar to ’w’, but also allows reading. It still truncates the file if it exists.

2. To ”append data to the end of an existing file,” the `'a'` mode is the correct and safest choice.

Final Answer: The mode used for appending is ’a’.

Answer: (C)

Q40.

Solution

Concept:

Command-line arguments are parameters passed to a script when it is executed from a terminal or command prompt. For example, in the command `python script.py data.txt`, ”data.txt” is an argument. Python provides access to these arguments through the `sys` module, which acts as an interface between the Python interpreter and the operating system.

Solution:

1. The `sys.argv` is a list in Python that contains all the command-line arguments passed to the script.
2. Let’s look at its structure:
- `sys.argv[0]`: The first element is always the name of the script itself.
- `sys.argv[1:]`: These are the additional arguments provided by the user.

3. This is useful for writing scripts that need to process different files or behave differently based on user input at the time of execution.
4. It does not store environment variables (that’s `os.environ`), system exceptions, or general global variables.
5. Thus, `sys.argv` is dedicated to command-line input.

Final Answer: It is used to store command-line arguments.

Answer: (B)

Q41.

Solution

Concept:

Python is a dynamically typed language, meaning the type of a variable is determined at runtime. However, Python is also "strongly typed," which means it does not allow operations between incompatible data types without explicit conversion. When the interpreter encounters an operation that is not supported for the given combination of object types—such as attempting to perform arithmetic on a string and an integer—it stops execution and raises a specific exception to signal this mismatch.

Solution:

1. Let's analyze the common exceptions provided in the options:

- **ValueError**: Occurs when a function receives an argument of the correct "type" but an inappropriate "value" (e.g., trying to convert `int("abc")`).
- **TypeError**: This is raised when an operation or function is applied to an object of an inappropriate type. Adding a string to an integer (`'5' + 5`) is the classic example of this.
- **NameError**: Raised when a local or global name (variable) is not found.
- **SyntaxError**: Raised when the Python parser encounters a breach in the rules of the language structure.

2. In the specific case of "adding a string to an integer," Python cannot decide whether to perform concatenation or addition, so it raises a `TypeError`.

3. To fix this, the programmer must explicitly convert one type to the other using functions like `int()` or `str()`.

Final Answer: The error that occurs is TypeError.

Answer: (B)

Q42.

Solution

Concept:

Reading data from a file is a core part of file handling in Python. The `read()` method belongs to the file object and is used to retrieve content from an open file. Depending on the arguments passed to it, the method can read the entire file or just a specific portion. It is important to distinguish between reading "lines" (which is done by `readline()` or `readlines()`) and reading "units" (bytes or characters).

Solution:

1. The syntax of the method is `file_object.read(n)`.
2. Here, the parameter *n* is optional.
3. If *n* is not specified or is negative, the method reads and returns the entire content of the file.
4. If *n* is a positive integer, the method reads at most *n* units from the current file pointer position.

- In **Text Mode** (`'r'`), it reads *n* characters.
- In **Binary Mode** (`'rb'`), it reads *n* bytes.

5. Since characters are essentially represented as bytes in memory, the most accurate general description is that it reads *n* bytes/characters.
6. It does not read *n* lines; to read lines, one would use `f.readlines(n)` or a loop.

Final Answer: The read(n) method reads n bytes/characters.

Answer: (B)

Q43.

Solution

Concept:

In Python's `try...except` block, you can handle different types of exceptions specifically (e.g., `except ZeroDivisionError:`). However, it is impossible to predict every single error that might occur. To make a program robust and prevent it from crashing due to an unhandled error, a "catch-all" block can be used. This block is placed at the end of the exception hierarchy to capture any exception that did not match the specific ones listed above it.

Solution:

1. Let's look at the structure of an exception handler:

- `try`: Contains the code that might cause an error.
- `except SpecificError`: Handles a particular error.
- `except Exception`: This is the base class for almost all Python exceptions. By catching `Exception`, you catch any error that inherits from this class.

2. The block `except Exception as e:` (or simply `except:`) acts as a safety net for any uncaught exceptions.

3. Let's check other blocks:

- `finally`: Runs no matter what (even if no exception occurred).
- `else`: Runs only if "no" exception occurred in the try block.
- `try`: Is the starting block, not the handler.

4. Therefore, `except Exception:` is the correct way to catch any remaining exceptions.

Final Answer: The block is except Exception:.

Answer: (A)

Q44.

Solution

Concept:

Switching techniques determine how data is routed across a network between the sender and the receiver. The three primary types are Circuit Switching (used in traditional telephony), Message Switching, and Packet Switching. The modern Internet is designed as a decentralized network where data is broken down into smaller units to ensure efficiency and reliability, even if some parts of the network fail.

Solution:

1. The Internet utilizes **Packet Switching**.
2. In Packet Switching:

- Data is broken into small pieces called ”packets.”
- Each packet contains the destination IP address and a sequence number.
- Packets can take different paths to reach the same destination based on the current network traffic.
- At the destination, the packets are reassembled into the original message.

3. This is different from **Circuit Switching**, which requires a dedicated physical path for the entire duration of the communication (like an old landline call).
4. Packet Switching is much more efficient because it allows many users to share the same network resources simultaneously without needing dedicated lines.

Final Answer: The Internet uses Packet Switching.

Answer: (B)

Q45.

Solution

Concept:

Malware (Malicious Software) is categorized based on how it behaves, replicates, and spreads. While all malware is harmful, some types require a ”carrier” or ”host” to exist, much like a biological virus. Others are more sophisticated and function as independent programs that can traverse the network and infect other systems without any human interaction or host file.

Solution:

1. Let's define the types of malware mentioned:

- **Virus**: A piece of code that attaches itself to a legitimate program or file (host). It cannot spread without the host file being executed or shared.
- **Worm**: A standalone piece of malware that replicates itself in order to spread to other computers. It uses the network to send copies of itself to other systems without needing a host file or human intervention.
- **Trojan**: Disguises itself as legitimate software but performs malicious actions. It does not replicate itself like a virus or worm.
- **Logic Bomb**: Malicious code that is triggered by a specific event or time.

2. Because a Worm is a standalone application that spreads through vulnerabilities in the network, it does not need a host file.

Final Answer: A Worm does not need a host file to replicate.

Answer: (B)

Q46.

Solution

Concept:

Email communication involves different protocols for sending and receiving messages. While SMTP (Simple Mail Transfer Protocol) is used to push or send emails from a client to a server, receiving protocols are needed to pull those messages down to a user's device. There are two primary protocols for receiving: IMAP and POP3. The key difference lies in how they handle the storage and synchronization of messages between the server and the local machine.

Solution:

1. Let's analyze the protocols used in email handling:

- **SMTP**: Used for "sending" emails. It moves the mail from the sender's device to the mail server.
- **POP3 (Post Office Protocol version 3)**: This protocol is designed to "download" emails from the server to a local device. By default, once the emails are downloaded, they are deleted from the server. This makes it ideal for users who access mail from a single device.
- **IMAP (Internet Message Access Protocol)**: This protocol allows you to "view" emails on the server. It syncs the messages across multiple devices.
- **HTTP/FTP**: These are general-purpose protocols for web browsing and file transfers, not specifically for mail retrieval.

2. The question specifically mentions retrieving emails and "storing them on a local device," which is the classic behavior of POP3.

Final Answer: The protocol is POP3.

Answer: (B)

Q47.

Solution

Concept:

In network security, storing passwords in plain text is a critical vulnerability. To protect user data, passwords are ”hashed” using cryptographic algorithms (like SHA-256). However, if two users have the same password, they will have the same hash, making the system vulnerable to ”Rainbow Table” attacks (pre-computed tables of hashes). To prevent this, security experts use a technique called ”Salting” to ensure every hash is unique, even for identical passwords.

Solution:

1. **Salting** is the process of adding a unique, random string of characters (the salt) to a user's clear-text password before it is passed through a hashing function.
2. Even if two users choose ”password123”, their salts will be different (e.g., Salt A and Salt B).
3. Consequently:
- Hash(password123 + Salt A) ≠ Hash(password123 + Salt B).
4. This ensures that:
- An attacker cannot use a single pre-computed table to crack passwords for the entire database.
- If an attacker sees two identical hashes, they still won't know if the underlying passwords are the same.
5. It is a fundamental practice in modern authentication systems to enhance password security against brute-force and dictionary attacks.

Final Answer: Salting is adding random data to passwords before hashing.

Answer: (A)

Q48.

Solution

Concept:

The OSI (Open Systems Interconnection) model is a conceptual framework used to understand network interactions in seven distinct layers. Each layer has a specific responsibility. While lower layers (1-3) handle the physical movement of data across the network and routing, the middle layer acts as a bridge to ensure that the data sent by the application is received correctly and in the right order by the destination.

Solution:

1. Let's look at the layers in the options:

- **Network Layer (Layer 3)**: Responsible for routing packets across different networks using IP addresses.
- **Transport Layer (Layer 4)**: This is the first "end-to-end" layer. Its primary duties include flow control, segmentation/desegmentation, and error recovery (retransmitting lost packets). Protocols like TCP operate here. This matches the question's description perfectly.
- **Session Layer (Layer 5)**: Manages the start, stop, and restart of sessions between applications.
- **Data Link Layer (Layer 2)**: Handles node-to-node data transfer and error detection at the local hardware level (MAC addresses).

2. The Transport Layer is considered the "heart" of the OSI model because it ensures the reliability of the communication channel between the source and destination hosts.

Final Answer: The Transport Layer is responsible for end-to-end communication.

Answer: (B)

Q49.

Solution

Concept:

The HTTP protocol, which powers the web, is ”stateless.” This means that every time a web browser requests a page, the server treats it as a completely new interaction and has no memory of previous requests. To create a seamless experience—such as staying logged in or keeping items in a shopping cart—web developers use small text files that the server sends to the browser to be stored locally. These are called Cookies.

Solution:

1. Cookies are used primarily to **maintain state information**.
2. Without cookies, a user would have to log in every time they clicked a new link on a website because the server would ”forget” who they are.
3. Functionalities of cookies include:
- **Session Management**: Keeping users logged in or remembering site preferences.
- **Personalization**: Showing content or ads based on previous browsing behavior.
- **Tracking**: Analyzing how users interact with a site.
4. They do not store passwords permanently on the server (they are stored on the client), they do not increase CPU speed, and they are not an antivirus tool.

Final Answer: Cookies are used to maintain state information across requests.

Answer: (B)

Q50.

Solution

Concept:

Cryptography is the practice of securing information by transforming it into a format that is unreadable to unauthorized parties. The core of this process involves using mathematical algorithms and ”keys.” There are two main directions in this process: turning readable information into a secret code, and turning that secret code back into readable information.

Solution:

1. Let’s define the terms related to secure communication:
- **Plain Text**: The original, readable message (e.g., ”Hello”).
- **Cipher Text**: The scrambled, unreadable version of the message.
- **Encryption**: The process of converting plain text into cipher text using an algorithm and a key. This is what the question describes.
- **Decryption**: The reverse process—converting cipher text back into plain text.
- **Encoding**: Converting data into a different format (like Base64 or binary) for transmission; it is not necessarily for security and usually doesn’t involve a secret key.
2. Encryption is the fundamental building block of modern digital security, used in everything from HTTPS websites to encrypted messaging apps like WhatsApp.

Final Answer: The process is called Encryption.

Answer: (C)

Answer Key

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