Match the following programming concepts in C++ with their most accurate descriptions:
1. Tokens, Identifiers, Variables, and Constants 2. Overloading, Inheritance, Templates, Exception and Event Handling 3. Streams and Files, Multifile Programs 4. Control Statements, Functions Parameter Passing, Virtual Functions, Class and Objects
Choose the correct answer from the options given below:
(1) 1-A, 2-D, 3-B, 4-C
(2) 1-C, 2-B, 3-D, 4-A
(3) 1-D, 2-C, 3-A, 4-B
(4) 1-A, 2-B, 3-D, 4-C
Answer Key: 2
Solution:
- Tokens, Identifiers, Variables, and Constants: These are the fundamental building blocks in C++ used to define syntax elements and data.
- Overloading, Inheritance, Templates, Exception and Event Handling: These are advanced features facilitating code reuse, runtime behavior, and exception safety.
- Streams and Files, Multifile Programs: These relate to input/output handling, file operations, and organizing large programs across multiple files.
- Control Statements, Functions Parameter Passing, Virtual Functions, Class and Objects: These are core concepts related to program flow control, parameter management, polymorphism, and object-oriented programming.
Hence, Option (2) correctly matches the topics with their descriptions.

2. Match the following design techniques with their operational strategies:
1. Divide and Conquer 2. Dynamic Programming 3. Greedy Algorithms 4. Backtracking
Choose the correct answer from the options given below:
(1) 1-B, 2-C, 3-D, 4-A
(2) 1-A, 2-D, 3-C, 4-B
(3) 1-C, 2-A, 3-B, 4-D
(4) 1-D, 2-B, 3-A, 4-C
Answer Key: 2
Solution:
- Divide and Conquer: Breaks a problem into subproblems, solves each independently, and combines solutions.
- Dynamic Programming: Solves problems by storing solutions to overlapping subproblems for optimality.
- Greedy Algorithms: Makes the optimal choice at each step without reconsideration, aiming for a global optimum.
- Backtracking: Explores all possibilities by building incrementally and abandoning paths that fail constraints.
Hence, Option (2) accurately associates the techniques with their operational strategies.

3. Match the following cloud service models and IoT concepts with their defining features:
1. SaaS, PaaS, IaaS 2. Virtualization, Virtual Server, Cloud Storage, Database Storage 3. Resource Management, Service Level Agreement, Basics of IoT 4. Public Cloud, Private Cloud
Choose the correct answer from the options given below:
(1) 1-D, 2-C, 3-B, 4-A
(2) 1-A, 2-D, 3-C, 4-B
(3) 1-B, 2-A, 3-D, 4-C
(4) 1-C, 2-B, 3-A, 4-D
Answer Key: 3
Solution:
- SaaS, PaaS, IaaS: These are cloud service models with increasing levels of control and abstraction.
- Virtualization, Virtual Server, Cloud Storage, Database Storage: These relate to resource abstraction, flexible allocation, and data management in cloud computing.
- Resource Management, Service Level Agreement, Basics of IoT: Key aspects for maintaining quality,

resource allocation, and foundational IoT understanding.
- Public Cloud, Private Cloud: Deployment models indicating accessibility, security, and ownership.
Hence, Option (3) correctly matches the cloud computing and IoT concepts with their features.

Match the following digital logic components with their respective Boolean simplifications or functions:
1. Logic Gate Expression/Function
I. NAND A. Produces high output only when all inputs are high
II. XOR B. Output is true when an odd number of inputs are true
III. NOR C. Outputs true only when all inputs are false
IV. AND D. Outputs true only when all inputs are true
Choose the correct answer from the options given below:
(1) I-D, II-B, III-C, IV-A
(2) I-A, II-C, III-D, IV-B
(3) I-B, II-A, III-C, IV-D
(4) I-C, II-D, III-A, IV-B
Answer Key: 3
Solution:
? NAND: The negation of AND, which outputs false only when all inputs are true, thus its Boolean expression is (AB)'.
? XOR: Outputs true when an odd number of inputs are true; its Boolean function is A?B.
? NOR: Outputs true only when all inputs are false; its Boolean expression is (A+B)'.
? AND: Outputs true only when all inputs are true; Boolean expression is AB.
Matching these, option (3) correctly aligns each component with its function.
Hence, Option (3) is the right answer.

2. Match the following counting principles and probability concepts with their descriptions:
A. Pigeonhole Principle B. Mathematical Induction C. Bayes? Theorem D. Inclusion-Exclusion Principle
1. States that if n objects are placed into m boxes, and n > m, then at least one box contains more than one object.
2. A method for proving the truth of an infinite sequence of propositions by verifying a base case and an inductive step.
3. Provides a way to update the probability estimate for an event based on new evidence.
4. Calculates the size of the union of multiple sets by subtracting overlaps.
Choose the correct answer from the options given below:
(1) A-2, B-1, C-3, D-4
(2) A-1, B-2, C-4, D-3
(3) A-1, B-2, C-3, D-4
(4) A-2, B-1, C-4, D-3
Answer Key: 2
Solution:
? Pigeonhole Principle (A): Asserts that placing n objects into m boxes with n > m guarantees at least one box contains multiple objects.
? Mathematical Induction (B): A proof technique establishing the truth for all natural numbers via base case and inductive step.
? Bayes? Theorem (C): Updates prior probabilities based on new evidence, enabling posterior probability calculation.
? Inclusion-Exclusion (D): A combinatorial formula for calculating the union of multiple sets by accounting for overlaps.
Matching these descriptions, option (2) correctly pairs each concept.
Hence, Option (2) is the right answer.

3. Match the following control unit design aspects with their descriptions:
1. Control Memory A. Stores the microinstructions governing control signals

2. Address Sequencing B. Determines the order of microinstructions execution
3. Design of Control Unit C. Involves creating logic circuits for generating control signals
Choose the correct answer from the options given below:
(1) 1-C, 2-B, 3-A
(2) 1-B, 2-C, 3-A
(3) 1-A, 2-B, 3-C
(4) 1-C, 2-A, 3-B
Answer Key: 3
Solution:
? Control Memory: Stores microinstructions that specify control signals at each step.
? Address Sequencing: Manages the order in which microinstructions are fetched and executed.
? Design of Control Unit: Involves designing the logic circuits such as finite state machines to generate control signals based on microinstructions.
Matching these, option (3) correctly aligns each aspect.
Hence, Option (3) is the right answer.

1. Consider a graph representing a network of cities connected by roads, where each edge has an associated travel time. You are to design an algorithm that can efficiently determine the minimum travel time path between any two cities, considering that some roads may have variable travel times depending on traffic conditions which change periodically. How should the graph algorithm be adapted to account for these temporal variations, and which approach would guarantee the optimal path in real-time?

Answer Key: (2)

Solution:
Option (2) is correct: To adapt traditional shortest path algorithms like Dijkstra?s for time-dependent edge weights, the algorithm must incorporate the temporal information into the edge weights, effectively making them functions of departure time. This leads to a time-dependent shortest path problem, often solved via algorithms like the Time-Dependent Dijkstra or more advanced methods such as the Time-Expanded Graph approach. These algorithms consider the periodic variation in travel times and dynamically update path costs based on current traffic conditions, thereby guaranteeing the optimal path in real-time. This approach ensures that the computed route reflects the current traffic pattern, providing a truly optimal solution under variable conditions.

Option (1) is incorrect: Using a standard Dijkstra?s algorithm without modification ignores the temporal variations and may yield suboptimal paths during peak traffic periods.

Option (3) is incorrect: Applying a simple Breadth-First Search (BFS) is ineffective because BFS does not handle weighted edges, especially when weights vary with time.

Option (4) is incorrect: Employing the Bellman-Ford algorithm without modifications does not inherently account for time-dependent weights, and its complexity makes it less suitable for real-time adjustments in large networks.

Hence, the correct answer is Option (2).

2. In a computer graphics rendering pipeline, a scan line polygon fill algorithm is used to fill complex polygons. Suppose you are given a polygon with multiple non-convexities and overlapping edges. Explain how the scan line algorithm must be modified to correctly fill such polygons without gaps or overlaps, especially considering the handling of intersections and boundary pixels. Which data structures and steps are critical to ensure an accurate fill?

Answer Key: (3)

Solution:
Option (3) is correct: For non-convex polygons with overlapping edges, the scan line fill algorithm must implement an active edge table (AET) that maintains all intersections of the current scan line with polygon edges, sorted by their x-coordinate. At each scan line, intersections are computed, and pairs of intersections are used to fill between them. To handle overlaps and non-convexities, the algorithm must also incorporate parity (odd-even) or non-zero winding rules to determine inside versus outside regions, which prevents gaps or overfilling. Critical steps include updating the AET as the scan line progresses, sorting intersections, and applying the fill rule. Proper handling of boundary pixels ensures the fill is precise, avoiding overlaps or gaps.

Option (1) is incorrect: Using only boundary fill methods ignores the need for managing multiple intersections and complex polygon structures, making it unsuitable.

Option (2) is incorrect: The midpoint circle or ellipse algorithms are specific to circle and ellipse drawing, not polygon fill, and do not address complex polygon filling issues.

Option (4) is incorrect: Flood fill algorithms are recursive or stack-based approaches suited for filling connected regions but do not inherently manage complex polygon intersections accurately.

Hence, the correct answer is Option (3).

3. Given an object-oriented programming system where classes can inherit from multiple parent classes, and considering that method overriding can cause ambiguity, analyze how the language?s compiler can resolve method calls when an object references a method present in multiple parent classes with different implementations. What is the most effective method of disambiguation, and how does it impact the design of class hierarchies?

Answer Key: (2)

Solution:
Option (2) is correct: The most effective method is to use a well-defined method resolution order (MRO), such as C3 linearization, which establishes a linear sequence of classes to search for the method, thereby resolving ambiguity systematically. This approach ensures that, when a method is called, the compiler follows the MRO to determine which implementation to invoke, maintaining consistency and predictability. Implementing MRO impacts class hierarchy design by enforcing a linear, unambiguous lookup path, encouraging careful planning of inheritance structures to prevent conflicts and ensure clarity in method resolution.

Option (1) is incorrect: Static binding of method calls at compile time cannot resolve ambiguity when multiple parent classes define the same method, especially in dynamic contexts.

Option (3) is incorrect: Relying solely on the most recently inherited class (rightmost class) for method resolution can lead to unpredictable behavior and is not a robust solution.

Option (4) is incorrect: Using only encapsulation without method resolution strategies does not address method ambiguity across multiple inheritance paths.

Hence, the correct answer is Option (2).