--Question Starting--
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).

--Question Starting--
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).

--Question Starting--
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).