--Question Starting--  
40. Consider an encryption scheme where a large prime number is essential for generating public and private keys. Suppose a developer needs to implement an algorithm that checks whether a given number, n, is prime. The chosen algorithm should efficiently handle very large numbers typical in modern cryptographic applications. Which algorithm is most appropriate for this scenario given its efficiency and practicality in cryptographic contexts?  
(1) Miller-Rabin primality test  
(2) Sieve of Eratosthenes  
(3) Euclid's algorithm for computing greatest common divisors  
(4) Trial division method  
Answer Key: 1  
Solution:  
• (Correct): The Miller-Rabin primality test is a probabilistic test that provides a high level of accuracy for large numbers and is commonly used in cryptographic applications due to its balance of efficiency and reliability.  
• (Incorrect): While effective for smaller ranges, the Sieve of Eratosthenes is not practical for very large numbers due to its memory and processing requirements.  
• (Incorrect): Euclid's algorithm is used for finding greatest common divisors, not for testing primality.  
• (Incorrect): Trial division is simple but extremely slow for large numbers, making it impractical for use in cryptography.  
Hence, Option (1) is the right answer.  
  
--Question Starting--  
35. An algorithm designer is tasked with optimizing a resource allocation system where multiple tasks with varying priorities must be assigned to a limited number of processors to achieve the best throughput. The tasks can arrive dynamically, and their processing can be paused and resumed. Which design technique will most effectively solve this complex, real-time scheduling problem?  
(1) Greedy algorithm  
(2) Dynamic programming  
(3) Backtracking  
(4) Branch and bound  
Answer Key: 2  
Solution:  
• (Incorrect): Greedy algorithms make locally optimal choices and might not adapt well to dynamic changes in task prioritization or processor availability.  
• (Correct): Dynamic programming can effectively handle the complexities of dynamic task arrival and the need for optimal state reassessment after every new task arrival or task completion, making it ideal for achieving the best overall throughput in a real-time system.  
• (Incorrect): Backtracking is typically used for decision problems where all possible solutions need to be explored; it is inefficient for real-time systems.  
• (Incorrect): Branch and bound is used to find the optimal solution for combinatorial problems and would be less efficient in a dynamic and real-time context.  
Hence, Option (2) is the right answer.  
  
--Question Starting--  
30. In a large social network, an algorithm is required to find the shortest path between two users to suggest the quickest route for information flow or friend recommendation. The network changes frequently with users joining and leaving, and paths can become obsolete quickly. Which graph algorithm is best suited for finding shortest paths in such a dynamic graph?  
(1) Dijkstra’s algorithm  
(2) Bellman-Ford algorithm  
(3) Floyd-Warshall algorithm  
(4) A\* search algorithm  
Answer Key: 3  
Solution:  
• (Incorrect): Dijkstra’s algorithm is efficient for static graphs with non-negative weights but does not handle dynamic updates efficiently.  
• (Incorrect): Bellman-Ford handles negative weights but like Dijkstra’s, isn’t the best for frequent updates.  
• (Correct): Floyd-Warshall algorithm, despite being computationally intensive, calculates the shortest paths between all pairs of nodes simultaneously and can be quickly updated for incremental changes, making it suitable for networks where paths need constant updates.  
• (Incorrect): A\* is great for single-source shortest paths with heuristics to guide the search but less effective for dynamic, all-pairs requirements.  
Hence, Option (3) is the right answer.  
  
--Question Starting--  
45. A software company is developing a new file management system to be integrated into the latest version of a popular operating system. The system needs to efficiently handle a large volume of small to large files and support both local and networked user access. Which component of the Windows operating system architecture should be most intensively focused on to optimize for these requirements?  
(1) File System  
(2) Terminal Services  
(3) Memory Management  
(4) Device Management  
Answer Key: 1  
Solution:  
• (Correct): The File System is directly related to how data is stored, accessed, and managed. Optimizing the file system will directly impact the performance and efficiency of the file management system, especially in handling various file sizes and types, both locally and over a network.  
• (Incorrect): Terminal Services deal with remote desktop access and session management, which, while important, are less directly impactful on file management system performance.  
• (Incorrect): Memory Management is crucial for overall system performance but is less specific to the nuances of file system operations.  
• (Incorrect): Device Management primarily handles the interaction of hardware devices with the OS, which is tangential to file handling efficiency.  
Hence, Option (1) is the right answer.  
  
--Question Starting--  
50. An algorithmic research team is analyzing the performance of a new sorting algorithm. The initial tests show a recurrence relation T(n) = 2T(n/2) + n with a base case of T(1) = 1. They need to determine the asymptotic complexity of this algorithm to compare it with established sorting algorithms. What is the asymptotic complexity of this algorithm?  
(1) O(n log n)  
(2) O(n^2)  
(3) O(log n)  
(4) O(n)  
Answer Key: 1  
Solution:  
• (Correct): The recurrence relation T(n) = 2T(n/2) + n falls into the case 2 of the Master Theorem, where the work done at each level of the recursion forms a geometric series. Solving this recurrence gives a complexity of O(n log n), which is typical for divide-and-conquer sorting algorithms like merge sort.  
• (Incorrect): O(n^2) would be typical of algorithms with quadratic complexity, such as bubble sort, which does not match the given recurrence.  
• (Incorrect): O(log n) is typical of algorithms that decrease the problem size exponentially without significant work at each step, unlike the given relation.  
• (Incorrect): O(n) would indicate a linear complexity, which underestimates the work done in the recurrence, as it involves recursive division of the problem.  
Hence, Option (1) is the right answer.