Data Structure Scenario (15 MCQ)

Time Complexity, Stack, Queue, Circular Q, Priority Q, LinkedList, Sorting, Searching

Points: 2/15

1. Enter your KU ID *

   cc026

2. Enter your Name *

   Sophea

3. Enter your Course (Cloud/Software/Security) *

   Cloud

✕ **Incorrect** 0/1 Points

4. You are developing a system to manage incoming requests to a web server. The system needs to efficiently handle a large number of requests while ensuring fairness among all requests. Additionally, the system should be able to reuse empty slots in the queue. Which data structure would be most suitable for implementing the request management system in this scenario? *

   - ○ Stack
   - ○ Circular Queue
   - ◉ Priority Queue

- Queue

✗ **Incorrect** 0/1 Points

5. Which of the following statements about merge sort is true *

- (•) Merge sort has a time complexity of O(n^2)
- ( ) Merge sort is an in-place sorting algorithm.
- ( ) Merge sort uses the divide-and-conquer strategy
- ( ) Merge sort is a stable sorting algorithm.

✓ **Correct** 1/1 Points

6. Which of the following statements about bubble sort is true *

- ( ) Bubble sort is an efficient sorting algorithm for large datasets
- ( ) Bubble sort has a time complexity of O(n log n).
- (•) Bubble sort repeatedly swaps adjacent elements if they are in the wrong order.
- ( ) Bubble sort is a stable sorting algorithm

✗ **Incorrect** 0/1 Points

7. You are given an array of integers of length n. You need to find the maximum and minimum elements in the array. Which of the following algorithms would have the best time complexity for this scenario? *

- ( ) Iterating through the array and comparing each element to find the maximum and minimum.
- ( ) Sorting the array in ascending order and then selecting the first and last elements.
- (•) Using a binary search algorithm to find the maximum and minimum.
- ( ) Creating a hash table and inserting all elements, then finding the maximum and minimum based on the hash table.

✕ **Incorrect** 0/1 Points

8. You are implementing a program to simulate a printer queue. Each print job is added to the queue, and the printer processes jobs in the order they were added. However, there is a priority print job that needs to be printed before all others. Which data structure would be most suitable for implementing the priority functionality in this scenario? *

- ○ Circular Queue
- ○ Priority Queue
- ◉ Linear Queue
- ○ Stack

✕ **Incorrect** 0/1 Points

9. In a singly linked list, which operation has a time complexity of O(n), where n is the number of nodes in the linked list? *

- ○ Insertion at the beginning
- ○ Insertion at the end
- ○ Deletion from the beginning
- ◉ Deletion from the end

✕ **Incorrect** 0/1 Points

10. You are given a task to implement a dictionary where each word is associated with its meaning. The dictionary should support efficient lookup of word meanings. Which data structure would you choose for this task? *

- ○ Array
- ◉ Linked List
- ○ Hash Table
- ○ Binary Search Tree

✕ **Incorrect** 0/1 Points

11. You are building an application that needs to efficiently store a collection of unique elements and perform fast search operations. Additionally, the order of insertion needs to be preserved. Which data structure would be most suitable for this scenario? *

- (●) Stack
- ( ) Queue
- ( ) Linked list
- ( ) Hash table

✕ **Incorrect** 0/1 Points

12. You are developing a music playlist application. Each song in the playlist contains information such as the song title, artist, and duration. Users should be able to add songs to the playlist and play them in the order they were added. Additionally, users should be able to remove songs from the playlist at any position. Which data structure would be most suitable for implementing the playlist in this scenario? *

- (●) Array
- ( ) Linked List
- ( ) Stack
- ( ) Queue

✕ **Incorrect** 0/1 Points

13. You are developing a system for a ticket counter at a theme park. The counter has a limited number of slots to serve customers. When a customer arrives, they join a queue and are assigned the next available slot. If all slots are occupied, the next customer in line should be assigned to the first available slot. Which data structure would be most suitable for implementing the ticket counter in this scenario? *

- (●) Stack
- ( ) Queue
- ( ) Circular Queue

- Priority Queue

✗ **Incorrect** 0/1 Points

14. You are implementing a system to process incoming requests from multiple clients. Each request needs to be handled in the order it arrives. Which data structure(s) would be most suitable for this scenario? *

- Stack
- Queue
- Array
- LinkedList

✗ **Incorrect** 0/1 Points

15. You are developing an algorithm to search for a specific number in a sorted array of integers. You want to ensure that the algorithm has the most efficient time complexity. Which algorithm would be most suitable for this scenario? *

- Linear Search
- Binary Search
- Bubble Sort
- Quick Sort

✓ **Correct** 1/1 Points

16. The five items: A, B, C, D, and E are pushed in a stack, one after other starting from A. The stack is popped four items and each element is inserted in a queue. The two elements are deleted from the queue and pushed back on the stack. Now one item is popped from the stack. The popped item is *

- A
- B

C

D

Incorrect 0/1 Points

17. You are implementing a ticketing system for a cinema. The system needs to manage the ticket sales in a fair and orderly manner. When a customer arrives, they join a queue to purchase tickets. However, there is a separate priority queue for members who have special VIP access, and they get served before the regular customers. Which data structure(s) would be most suitable for implementing the ticketing system in this scenario? *

- Stack for regular customers, Queue for VIP members
- Queue for regular customers, Stack for VIP members
- Queue for both regular customers and VIP members
- Stack for both regular customers and VIP members

Incorrect 0/1 Points

18. You are given an array of integers and need to determine if there are any duplicate elements in the array. Which of the following approaches has the most efficient time complexity for solving this problem? *

- Sorting the array and then comparing adjacent elements.
- Using nested loops to compare each element with every other element.
- Creating a hash set and checking for duplicate elements while inserting into the set.
- Using a binary search tree to store and search for duplicate elements.