Prince Sultan University
CCIS - Department of Computer Science

Major Exam 2
Term 242

Course title: Data Structures and Algorithms **Course Code**: CS 210

Exam date: 21/04/2025 **Exam Time: 50** minutes

Student Name:

Student ID:

Circle Instructor Name:

Circle Section Time:

Question Number	CLO	Question points	Score
Question 1	**CLO 1**	4	
Question 2	**CLO 3**	4	
Question 3	**CLO 4**	7	
Total out of		**15**	

Instructions:

- This exam contains three questions with multiple parts.
- Time allowed: 50 minutes
- Closed Book, Closed Notes.
- Use of Calculators and / or computing devices / smartphones etc is strictly prohibited.
- Answer the problems on the exam sheets only. No additional attachments would be accepted.
- If you need extra space use the back of a page.
- When the "time is over" is called, it is students' responsibility to submit his exam to the invigilator. Submitting completed exam 3 minutes after the "time is over" will incur a penalty of **5 points**.
- Do **NOT** use erasable pens!

Few gentle reminders:

- If you get stuck on some problem for a long time, move on to the next one.
- The ordering of the problems is somewhat related to their relative difficulty. However, the order might be different for you!
- You should be better off by first reading all questions and answering them in order of what you think is the easiest to the hardest problem.
- Keep the points distribution in mind when deciding how much time to spend on each problem.

Prince Sultan University
CCIS - Department of Computer Science

For all MCQ Questions, COPY your answers in this area only. DO NOT write answers anywhere else (it will not be graded).

1.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
2.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
3.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
4.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
5.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
6.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
7.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
8.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
9.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
10.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
11.	Ⓐ Ⓑ Ⓒ Ⓓ	**1 point**
12.	Ⓐ Ⓑ Ⓒ Ⓓ	**1 point**
13.	Ⓐ Ⓑ Ⓒ Ⓓ	**1 point**
14.	Ⓐ Ⓑ Ⓒ Ⓓ	**1 point**
15.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**
16.	Ⓐ Ⓑ Ⓒ Ⓓ	**0.5 point**

Question 1 [/ 4 Points]

Answer the following MCQs (each has 0.5 weight)

1. The steps for sorting an array [7, 3, 5, 2, 6, 4, 1, 8**] are given below. Which algorithm was used?**

7, 3, 5, 2, 6, 4, 1, 8
3, 7, 2, 5, 4, 6, 1, 8
2, 3, 5, 7, 1, 4, 6, 8
1, 2, 3, 4, 5, 6, 7, 8

A) Selection Sort	B) Insertion Sort	C) Merge Sort	D) Quick Sort

2. The steps for sorting an array of characters [b, d, a, k, c, f**] are given below. Which algorithm was used?**

b, d, a, k, c, f
a, b, d, k, c, f
a, b, d, k, c, f
a, b, c, d, k, f
a, b, c, d, f, k

A) Insertion Sort	B) Selection Sort	C) Merge Sort	D) Quick Sort

3. If an array has all elements equal, what is the runtime complexity of Selection Sort?

A) O (1)	B) O (n)	C) O (n log n)	D) O (n^2)

4. If all elements of an array are already sorted in ascending order, which algorithm would work fastest to sort this array in descending order?

A) Selection Sort	B) Insertion Sort	C) Merge Sort	D) Quick Sort

5. What is the runtime for Quicksort, which is applied to an array that is already sorted? Assume no shuffle was done before sorting.

A) O (1)	B) O (n)	C) O (n log n)	D) O (n^2)

6. Assume an array of size n is given. Which of the following algorithms requires extra space of size n to sort this array?

A) Selection Sort	B) Insertion Sort	C) Merge Sort	D) Quick Sort

7. An array of strings is given [“cab”, “cad”, “can”, “cap”, “car”, “cat”]. How many partitions are made if sorted using the quick-sort algorithm, note that the first element is selected as the pivot?

A) 2	B) 3	C) 4	D) 5

8. Merge sort and Quick sort algorithms both have an average case runtime of O(n log n). For large arrays (data sets), why does quicksort perform better (faster)?

A) Quick sort is an in-place algorithm (it does not need extra memory)	B) Quick sort requires fewer comparisons than merge sort	C) Quick sort is a non-recursive algorithm, which avoids the overhead of recursion	D) Quick sort uses a fixed pivot, which ensures balanced partitioning and efficient sorting

Question 2 [/ 4 Points]

9. Consider a stack implemented using an array of size n. What is the most efficient way to check for overflow (Full condition) before pushing? [0.5 mark]

A) Check if top == -1
B) Check if top == 0
C) Check if top == n – 1
D) Check if array[top] is null

10. You are given two empty stacks, S1 and S2. Consider the following pseudocode: [0.5 mark]

```
S1.push(5)
S1.push(3)
S2.push(S1.pop())
S1.push(7)
S2.push(S1.pop())
S2.push(S1.pop())
S1.push(9)
S1.push(S2.pop())
x = S2.pop()
y = S1.pop()
```

After executing the code, what are the final contents of S1 and S2 (from bottom to top), and the values of x and y?

A) S1: [9] S2: [3] x = 7, y = 5	B) S1: [9] S2: [3] x = 5, y = 7
C) S1: [9] S2: [3] x = 3, y = 5	D) S1: [5] S2: [9] x = 3, y = 7

11. A circular queue of size 5 (fixed array size) is implemented with the following structure: [1 mark]
The queue uses a front and rear index.
Initially, front = -1 and rear = -1, indicating an empty queue.
On enqueue, if the queue is not full, rear = (rear + 1) % size, and the new element is placed at queue[rear].
On dequeue, if the queue is not empty, front = (front + 1) % size.
Now, consider the following sequence of operations:

enqueue(10), enqueue(20), enqueue(30), dequeue(), enqueue(40), enqueue(50), dequeue(), enqueue(60), enqueue(70)

What are the final contents of the circular queue (from front to rear), and what is the state of the front and rear pointers?

A) Queue: [70, 20, 30, 40, 60]	front = 2,	rear = 1
B) Queue: [60, 70, 30, 40, 50]	front = 2,	rear = 1
C) Queue: [60, 70, __, 40, 50]	front = 3,	rear = 1
D) Queue: [60, 70, __, __, 50]	front = 4,	rear = 1

12. Which of the following Java methods correctly transforms a queue of integers from: [1 mark]
`[1, 2, 3, 4, 5, 6]` to `[3, 2, 1, 4, 5, 6]`?

A)
```
public void f(Queue<Integer> q) {
    Stack<Integer> s = new Stack<>();
    while (!q.isEmpty()) {
        s.push(q. dequeue ());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
}
```

B)
```
public void f(Queue<Integer> q) {
    int size = q.size();
    for (int i = 0; i < size / 2; i++) {
        q.add(q.remove());
    }
}
```

C)
```
public void f(Queue<Integer> q) {
    int size = q.size();
    for (int i = 0; i < size / 2; i++) {
        q.add(q.remove());
    }
    Stack<Integer> s = new Stack<>();
    for (int i = 0; i < size / 2; i++) {
        s.push(q.remove());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
}
```

D)
```
public void f(Queue<Integer> q) {
    int size = q.size();
    Stack<Integer> s = new Stack<>();
    for (int i = 0; i < size / 2; i++) {
        s.push(q.dequeue());
    }
    while (!s.isEmpty()) {
        q.add(s.pop());
    }
    for (int i = 0; i < size - size / 2; i++) {
        q.add(q. dequeue ());
    }
}
```

13. You are implementing a queue using two stacks, [1 mark]
`s1` and `s2`, following the enqueue-heavy strategy.
The `enqueue()` operation performs the following steps:

1. **Move all elements from `s2` to `s1`**
2. **Push the new element to `s1`**
3. **Move all elements back from `s1` to `s2`**

What are the time complexities of `enqueue()` and `dequeue()` in this implementation?

A) enqueue() → O(1), dequeue() → O(n)

B) enqueue() → O(n), dequeue() → O(n)

C) enqueue() → O(1), dequeue() → O(1)

D) enqueue() → O(n), dequeue() → O(1)

Question 3 [/ 7 Points]

PART A: (/ 2 Points)

Given the fully parenthesized arithmetic expression, answer Q14 & Q15:

(((((a * (b + c)) - ((d - e) / f)) + ((g + h) * ((i - j) + (k / l)))) - (m + (n * (o - (p + q)))))

14. You are asked to construct a **binary expression tree** for the expression and analyze its structure and behavior. Which of the following statements is **correct**? **[1 mark]**

A) The root of the tree is a + operator, and the deepest subtree corresponds to (b + c).
B) A postorder traversal of the tree will evaluate (a * (b + c)) before evaluating (d - e) / f).
C) The expression tree has more operators at the leaves than operands.
D) The rightmost path from the root ends at the node containing operand a.

15. What would the **run time** be if the **post-order traversal** is applied to the expression? **[0.5 mark]**

A) O (1)
B) O (logn)
C) O (n)
D) O (nlogn)

16. The best-case runtime to delete the root of a Binary Search Tree is? **[0.5 mark]**

A) O (1)
B) O (logn)
C) O (n)
D) O (nlogn)

PART B: (/ 5 Points)

Given the following sequence of integers:

40, 20, 10, 30, 60, 50, 70, 65

A. **Insert** into an **empty Binary Search Tree (BST)**. [/ 1 mark]
B. **Delete** 20 from the tree, resulting in A, and indicate which case. [/ 0.5 mark]
C. **Insert** 25 from the tree resulting in B. [/ 0.5 mark]
D. **Delete** 60 from the tree resulting in C, and indicate which case. [/ 0.5 mark]
E. Give the **array representation** of the tree resulting in D. [/ 1 mark]

Construct a Binary Tree using the following traversal sequences: [/ 1.5 mark]

- **Pre-order:** M I N D S E T
- **Post-order:** N D I E T S M

What is the structure of the tree?

<End of Exam>