Prince Sultan University
CCIS - Department of Computer Science

Major Exam 1
Term 232

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

Exam date: 19/02/2024 **Exam Time: 50** minutes

Student Name:

Student ID:
Section #:

Question Number	CLO	Question points	Score
Question 1	**CLO 1**	5	
Question 2	**CLO 2**	7	
Question 3	**CLO 4**	3	
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 the 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 the 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.

Question 1 [5 points - CLO 1]

Part A: **(4 / points)**

Q1. Assume that you are given a **circular linked list** with sorted data (Integer values) in ascending order. Write a method **void insertSorted(int X)** that takes integer X as a parameter and inserts a new node containing this value at the appropriate location in the list. After the insertion, the sorted order should not change and the list remains sorted. Address all possible scenarios:

1. If the list is empty.
2. If the list has only 1 pre-existing node.
3. If the list has 2 or more nodes.

Also, give a **Big-Oh** notation for the worst-case runtime.

```
void insertSorted (int X){
      //create an new node and copy X
      Node N = new Node(X);
      //check if list is empty
      if(size==0){
          Cursor = N;
          N.next = Cursor;
          N.prev = Cursor;
      }
      else if (size==1)
      {
          Cursor.next =
          Cursor.prev = N;
          N.next = Cursor;
          N.prev = Cursor;
      }
      else
      {
          for(int i=0;i<size;i++)
          {
              if(Cursor.val> X)
                  break;
              Cursor = Cursor.next;
          }
          Node before = Cursor.prev;
          N.next=Cursor;
          N.prev=before;
          before.next=N;
          Cursor.prev=N;
      }
      size ++;
  }
```

Part B: **(1 / points)**

Give the output of the following code if the linked list has values (1, 2, 3, 4, 5, 6)

```
void fun2(Node head)
{
   if (head == null)
   {
      return;
   }
   System.out.print(head.data + " ");

   if (head.next != null)
   {
      fun2(head.next.next);
   }
   System.out.print(head.data + " ");
}
```

1 3 5 5 3 1

Question 2 [7 points - CLO 2]

Part A: **(4 / points)**

1	What advantage does a doubly linked list offer over a singly linked list? A) Lower memory consumption B) Simpler implementation C) Faster traversal D) Bidirectional traversal	**D**
2	Which operation is more efficient in a doubly linked list than a singly linked list? A) Insertion at the end B) Deletion in the middle C) Traversal from the beginning to the end D) Searching for a specific element	**A**
3	What is the space complexity of a doubly linked list with 'n' elements? A) O(n) B) O(1) C) O(log n) D) O(n^2)	**A**
4	In a circular linked list, how is the end of the list determined? A) By a NULL pointer B) By a specific end-of-list marker C) By a sentinel node D) By the node whose next pointer points to the first node	**D**

5	What happens if a circular linked list has only one node? A) It becomes a linear linked list B) It remains circular with the single node pointing to itself C) It becomes a doubly linked list D) It forms a loop without any nodes.	**B**
6	What is the time complexity for adding an element to the end of a Singly Linked List? A) O(1) I think this is true as well B) O(n) C) O(log n) D) O(n^2)	**B**
7	What is the runtime for searching an element in a Circular list that does not exist? A) O(n) B) O(log n) C) O(n log n) D) O(n^2)	**A**
8	Which operation cannot be performed on a circular linked list but can be performed to a singly linked list? A) Traversal B) Deletion C) Insertion at the end D) Searching for a specific element	**C**

Find the estimate for T(n) and the Asymptotic complexity (Big O) for the following code snippets by computing the number of primitive operations:

	Code Snippet	**T(n)**	**Big-O characterization**

1	`public Node printFirst(SinglyList S)`
	`{`
	`Node Temp = S.Head;`
	`while(Temp!=S.Head)`
	`  Temp = Temp.next;`
	`return Temp;`
	`}`
			O(1)

2	`int sum = 0;`
	`for (i = 1; i <= n; i += 2) {`
	`  for (j = 1; j <= n; j *= 2) {`
	`  sum = sum + i`
	`  }`
	`}`
			O(n log n)

3	`public int accumelate(int n){`
	`  int acc = 0;`
	`  for (int i = 1; i <= n; i++)`
	`  for (int j = 1; j <= n; j++)`
	`  acc = acc + i + j ;`
	`  return acc;`
	`}`
	`public void main(){`
	`Scanner Key= new Scanner(System.in);`
	`int n=Key.nextInt();`
	`for (i = 1; i <= n; i++)`
	`  System.out.println(accumelate(n));`
	`}`
			$O(n^3)$

4	`int k= 50;`
	`int sum = 0;`
	`for (i = 1; i <= k; i++) {`
	`  for (j = 1; j <= n; j++) {`
	`  sum = sum + i + j;`
	`  }`
	`}`
			O(k.n) or O(n)

Part B: **(1 / points)**

Give the best Big-O characterization for each of the following running time estimates (where n is the size of the input problem).

	Running time estimate	**Big-O characterization**
(a)	$2^{10} \log n + 2^n$	O(2
(b)	$3 \log n - 100 n^2 + 2^5 n$	$O(n^2)$
(c)	$99 \log n + 10 n - 77 n \log n$	O(n log n)
(d)	$8888^{27} + 2^{58}$	O(1)

Question 3 [3 points - CLO 4]

Part A: **(2 / points)**

```
int method1(int n)
{
        if (n == 1)
                return 1
        else
                return (n*n) + method1(n-1);
}
```

A. **What does the above method do? (0.5 point)**
B. **Draw the recursion trace for method1(5). (1.5 point)**
C. **What is big-O notation and T(n) of method1? (1 point)**

Add the squares of all value between n and 1

Method1(5)

5*5 + Method1(4)
4*4+Method1(3)
3*3 + Method1(2)
2*2 +Method1(1)
1

T(n) = 7n + c
O(n)

Part B: **(1 / point)**

Algorithms **A** and **B** spend exactly $T_A(n) = 7n^2\log_2 n$ and $T_B(n) = n^3$, respectively, for a problem of size n. Answer the following:

1. Find out the value of n_0 where algorithm A is better than algorithm B.
2. If the input size is 1000, which algorithm will you recommend to use?

$7n^2\log n < n^3$
$7\log n < n$
$7 < n/\log n$

You can select $n_0 = 64$, log 64 is 6 so 64/6 would be 10, i.e. > 7.

Obviously, A is always faster than B for any value of n>5. For n=1000, I will choose A.