King Saud University

College of Computer and Information Sciences

Department of Computer Science

CSC 212 Midterm 1 **Solution** – 1st Semester 2013

Date: November 14, 2012

Exam Duration: 2 hours

Student Name	
Student ID	

Guidelines:

1. Write your name and ID clearly on the top of each page.
2. Write your answer in the reserved space (do not use the leftmost column).

Question	Remarks	Mark
Question 1		
Question 2		
Question 3		
Question 4		
Total		

CSC 212 MT1	Name:	ID:

Question 1

Find the total number of primitive operations and Big Oh notation of the following methods.

a)

	Statements	S/E	Frequency	Total
1	void method1(int n)	0	-	-
2	{	0	-	-
3	int sum=0;	1	1	1
4	for(int i = 0; i <= 10; i = i+2) {	1	7	7
6	sum = sum + i;	1	6	6
7	}	0	-	-
8	System.out.println(sum);	1	1	1
9	}	0	-	-
	Total Operations →	15		
	Big Oh →	O(1)		

b)

	Statements	S/E	Frequency	Total
1	boolean method2(int n){	0	-	-
2	for(int i = 0; i < n; i++) {	1	n + 1	n + 1
3	for(int j = 1; j < n; j++) {	1	n.n	n^2
4	System.out.println(i + ": " + j);	1	n (n - 1)	n^2 - n
5	}	0	-	-
6	}	0	-	-
7	return true;	1	1	1
8	}	0	-	-
	Total Operations →	$2n^2 + 2$		
	Big Oh →	O(n^2)		

Question 2

Consider the following linked implementation of queues

```
public class Queue <T> {
   private Node<T> head, tail;
   private int size;
   public Queue() {
      head = tail = null;
      size = 0;
   }
:
.
}
```

a) Write down a member method that inserts a new element into the queue. Use the following header

```
public void enqueue (T e){
    if(head == null) {
      head = tail = new Node<T>(e);
    }
    else {
      tail.next = new Node<T>(e);
      tail = tail.next;
    }
    size++;
}
```

b) Write down a member method that returns the length (number of nodes) in the queue.

```
public int length(){
    return size;
}
```

c) Write down a member method that removes an element from the queue. The method must also return the data value that has been removed. Use the following header

```
public T serve(){

    T x = head.data;
    head = head.next;
    size--;
    if(size == 0)
      tail = null;
    return x;
}
```

d) Write down a member method, peek, that returns the data value of the first element in the queue without deleting it.

```
public T peek(){

    return head.data;

}
```

Question 3

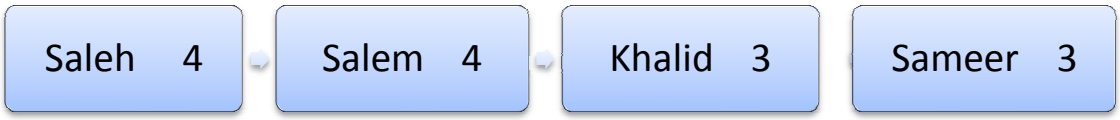
a) Write an efficient static (client/user) method that splits a queue of n elements into two queues. The elements with odd orders (i.e. 1st ,3rd, 5th, …) should be put in the first queue and elements with even orders (i.e. 2nd, 4th, 6th) should be put in the second queue.

```
public static void split(Queue<T> q, Queue<T> q1, Queue<T> q2){
  while(q.length()>0){
    T x=q.serve();
    q1.enqueue(x);
    if(q.lenght()!=0){
        T x=q.serve();
        q2.enqueue(x);
    }
} //while
}

A more readable alternative solution

public static void split(Queue<T> q, Queue<T> q1, Queue<T> q2){
 int i=1;
  while(q.length()>0){
    T x=q.serve();
    If(i%2=1)
        q1.enqueue(x);
    else
        q2.enqueue(x);
   i++;
    }
} //while
}
```

b) Suppose that the graph below represents the linked priority queue, q where numbers represent the priority of nodes.



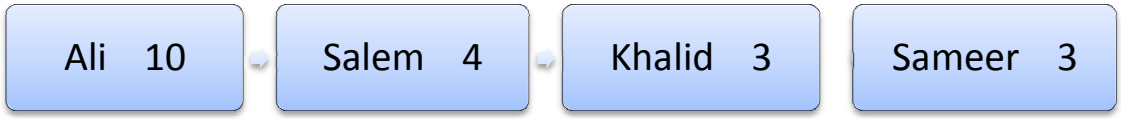
Redraw the queue showing the state of the private instance variables of q, after executing each of the following operations:

1) q.serve();

Answer:

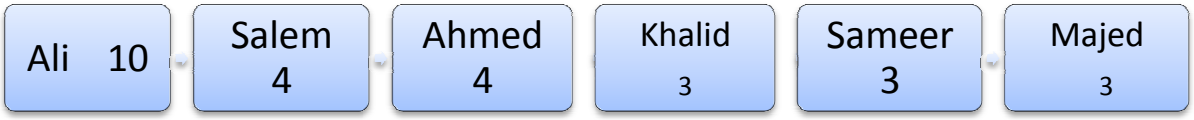
2) q.enqueue("Ali",10);

Answer:



3) q.enqueue("Ahmed", 4)

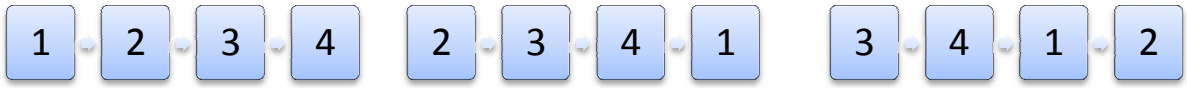
4) q.enqueu("Majed", 3)



Question 4

A circular left shift of a list consists in moving the first element to the last position while leaving the order of the remaining elements unchanged. For example:

L: L after one circular left shift: L after two circular left shifts:



a) Write a static method **CLS** (user the ADT) that takes as input a **non-empty** linked list **l** and an integer **n** (n >=0) and applies n circular left shifts to the list l.

```
public static void CLS(LinkedList<T> l, int n)
{
  T tmp;

  for(int i=0; i<n; i++){

        l.findFirst();     // Go to the first element
        tmp= l.retrieve(); // get the data
        l.remove();        // then remove it

        if(!l.empty()){//If the list contained more than 1 element
              while(!l.last())  // Go to the end
                    l.findNext();
        }

        l.insert(tmp); // then insert

  }
```

b) Write the method **CLS** member of the class LinkedList that takes as input an integer **n** (n >=0) and applies n circular left shifts to the list. (**Do not call any method when writing this method.)**

```
public class LinkedList<T>{
    private Node<T> head;
    private Node<T> current;
```

```
    public void CLS(int n)
    {

      Node<T> tmp, saveHead;
```

```
      If((head === null) || (head.next == null)) // If the list
      is empty or has has only 1 element, then it will not change
            return;

      for(int i=0; i<n; i++){

            saveHead= head;  // save the first node
            head= head.next; // then remove it

            tmp= head;
            while(tmp.next != null)  // Go to the end
                  tmp= tmp.next;
            }

            tmp.next= saveHead; // then insert
            saveHead.next= null; // If we do not do this, the list
            becomes circular

      }

   }
```