CSC 212 Midterm 1 - Fall 2015

College of Computer and Information Sciences, King Saud University
Exam Duration: 90 Minutes

20/10/2015

Question 1 [30 points]

1. Complete the array implementation of the ADT *Queue*. Write down the methods: *public int length(), public boolean full(), public void enqueue(T e),* and *public T serve()*. Write the method *public T serveTail()*, which removes and returns the last element of the queue (assume that the queue is not empty).

```
public class ArrayQueue<T> {
        private int maxSize, size, head, tail;
        private T[] data;
        public ArrayQueue(int maxSize) {
                this.maxsize = maxSize;
                size = head = tail = 0;
                data = (T[]) new Object[maxSize];
        }
}
```

2. Write the method *public void crs(int k)*, member of the class *LinkedList*, which performs a circular right shift of the list with k elements. Assume that $0 < k < n$, where n is the length of the list.

   Example 1.1. *Given the list* $l : A \to B \to C \to D \to E$, l.crs(1) *results in* $E \to A \to B \to C \to D$, l.crs(2) *results in* $D \to E \to A \to B \to C$, l.crs(3) *results in* $C \to D \to E \to A \to B$.

Question 2 [35 points]

For each of the two following methods, write down the S/E, frequency and total for every line, the total number of steps of the method and its O notation.

1.

```
1 void func1(int n) {                                   //Answer
2         int k = 3, j = 5, sum = 0;                    //3
3         for (int i = 0; i < n; i++)                   //n+1
4                 for (j = 1; j <= k; j++) {            //4n
5                         sum = i + j;                  //3n
6                         System.out.println(sum);      //3n
7                 }
8 }                                       //Total 11n+4, O(n)
```

2.

```
1 public void func2(int n) {
2         for (int i = 0; i < n*n*n; i++) {
3                 System.out.println(i);
4                 for (int j = 2; j < n; j++) {
5                         System.out.println(j);
6                 }
7         }
8         System.out.println("Goodbye!");
9 }
```

Question 3 [35 points]

1. Write the method *isCloserToHead*, member of the class *DoubleLinkedList*. It returns true if current is closer to the first element than the last element, and false otherwise. Assume that the list is not empty. The current element should not change after calling the method. **Do not use any other methods or auxiliary data structures**. The method signature is *public boolean isCloserToHead()*.

   Example 3.1. *Assuming the list contains:* $A \leftrightarrow B \leftrightarrow C \leftrightarrow D \leftrightarrow E \leftrightarrow F \leftrightarrow G$. *If current is on B or C, then calling* isCloserToHead() *returns true. However, if current is on D or E, then calling* isCloserToHead() *returns false.*

2. As a user of the Queue ADT, write the method *inBothHalves* that receives a Queue q and an element e. The method returns true if e appears in both the first and second halves of q, false otherwise. The order of q must not change. Assume that the length of q is even. **Do not use any auxiliary data structures**. The method signature is *public <T>boolean inBothHalves(Queue<T>q, T e)*.

   Example 3.2. *If we call the method over the queue* A, B, C, A *and element 'A', it should return true. If we call it for element 'B', it should return false.*

   Answer

```
public <T> boolean InBothHalves(Queue<T> q, T e){
                boolean flag=false,flag2=false;
                int half=q1.length()/2;

                for(int i=0;i<q.length();i++){
                        T s=q.serve();
                        if(s.equals(e) && i<half)
                                flag=true;
                        if(flag && i>=half && s.equals(e))
                                flag2=true;

                        q.enqueue(s);
                }
                return flag2;
        }
```

Specification of ADT Queue

- enqueue (Type e): **requires**: Queue Q is not full. **input**: Type e. **results**: Element e is added to the queue at its tail. **output**: none.
- serve (Type e): **requires**: Queue Q is not empty. **input**: none. **results**: the element at the head of Q is removed and its value assigned to e. **output**: Type e.
- length (int length): **requires**: none. **input**: none. **results**: The number of elements in the Queue Q is returned. **output**: length.
- full (boolean flag): **requires**: none. **input**: none. **results**: If Q is full then flag is set to true, otherwise flag is set to false. **output**: flag.