King Saud University	College of Computer and Information Sciences Department of Computer Science
Data Structures CSC 212	**Midterm Exam 1 Sample - Spring 2020**
Date: -	Duration: 90 minutes

Guidelines

•No calculators or any other electronic devices are allowed in this exam.

Student ID:	Name:
Section:	Instructor:

1	2	3.1	3.2	Total

Question 1 . . . . 20 points

Consider the function `f` below, member of `DoubleLinkedList`:

```
public void f(int n) {
  Node<T> p = head;
  for(int i = 0; i < n; i++) {
    if (p.next != null)
      p = p.next;
  }
  p.previous.next = p.next;
  if (p.next != null)
    p.next.previous = p.previous;
  p.next = head;
  p.next.previous = p;
  p.previous = null;
  head = p;
}
```

Choose the correct result in each of the following cases:

1. The list `l`: A, B, C, D, E, after calling `l.f(3)`, `l` becomes:

   Ⓐ B, C, D, E Ⓑ D, A, B, C, E Ⓒ E, B, C, D Ⓓ A, D, E, B, C Ⓔ None

2. The list `l`: A, B, C, D, E, after calling `l.f(1)`, `l` becomes:

   Ⓐ A, B, C Ⓑ E, A, B, C, D Ⓒ B, C, D, E, A Ⓓ B, A, C, D, E Ⓔ None

3. The list `l`: A, B, C, D, E, after calling `l.f(5)`, `l` becomes:

   Ⓐ A Ⓑ E, A, B, C, D Ⓒ A, B, C, D, E Ⓓ E, A, B, C, D Ⓔ None

4. The list `l`: A, B, C, D, E, after calling `l.f(2)`, `l` becomes:

   Ⓐ *empty* Ⓑ E, D, C, B, A Ⓒ C, A, B, D, E Ⓓ E, C, D Ⓔ None

Question 2 .... 20 points

Write the method **public static**`<T>` **void** `moveToEnd(List<T> l,` **int** `i)`, user of the ADT `List`. The method takes a list `l` and an index `i`. It moves the element at the `i`-th position to the end of the list. You can assume `i` to be within the list, and that the first element has the position 0.

Example 1. *If* `l`*:* $a \to c \to d \to b \to r \to x$*, then after calling* `moveToEnd(l, 2)`*,* `l` *will be:* $a \to c \to b \to r \to x \to d$*.*

Question 3 .... 60 points

We want to write an algorithm that stores data according to two priorities: `pr1`, where higher values have higher priority, and `pr2` where lower values have higher priority. In case of a tie, apply FIFO.

Example 2. *If we enqueue the following elements:* $(A, 2, 3), (B, 4, 2), (C, 4, 1)$*, then serve according to* `pr1` *returns* $(B, 4, 2)$*, and serve according to* `pr2` *returns* $(C, 4, 1)$.

To achieve this, we will use a data structure that supports the interface `PQ2`, which is defined as follows:

```
public interface PQ2<T> {
  // Returns the number of elements in the queue.
  int length();
  // Returns true if the queue is full.
  boolean full();
  // Add element e with priorities pr1 and pr2.
  void enqueue(T e, int pr1, int pr2);
  // Removes and returns the element with highest pr1 value.
  PQ2Elem<T> serve1();
  // Removes and returns the element with lowest pr2 value.
  PQ2Elem<T> serve2();
}
public class PQ2Elem<T> {
  public T data;
  public int pr1;
  public int pr2;
  public PQ2Elem(T data, int pr1, int pr2) {
    this.data = data;
    this.pr1 = pr1;
    this.pr2 = pr2;
  }
}
```

1. Write the method **public static** `<T> PQ2Elem<T> minMax(PQ2<T> q)`, which returns the element in `q` with the lowest `pr2` value among those who have the highest `pr1` value. The queue is assumed non-empty and must not change after the call. In case of a tie apply FIFO.

   Example 3. *If we enqueue the following elements:* $(A, 2, 3), (B, 4, 2), (C, 4, 1), (D, 3, 0)$*, then call* `minMax` *, the method returns* $(C, 4, 1)$.

   ....

   ....

   ....

   ....

   ....

   ....

   ....

   ....

   ....

   ....

2. Implement this interface (include any other classes you need). **Do not use any data structure in your implementation**.