Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

Introduction to Computing II (ITI 1121)
FINAL EXAMINATION: SOLUTIONS

Instructors: Guy-Vincent Jourdan and Marcel Turcotte

April 2018, duration: 3 hours

Identification

Last name: ______________________ First name: ______________________________

Student #: ___________ Seat #: ____ Signature: ____________________ Section: A or B or C

Instructions

1. This is a closed book examination.
2. No calculators, electronic devices or other aids are permitted.
   (a) Any electronic device or tool must be shut off, stored and out of reach.
   (b) Anyone who fails to comply with these regulations may be charged with academic fraud.
3. Write your answers in the space provided.
   (a) Use the back of pages if necessary.
   (b) You may not hand in additional pages.
4. Do not remove pages or the staple holding the examination pages together.
5. Write comments and assumptions to get partial marks.
6. Beware, poor hand-writing can affect grades.
7. Wait for the start of the examination.

Marking scheme

Question	Maximum	Result
1	5	
2	10	
3	5	
4	15	
5	10	
6	4	
7	6	
8	10	
Total	**65**	

Question 7 (6 marks)

Carefully analyze the following Java and answer the question on the next page.

```
public class LinkedList<E> {
    private static class Node<T> {
        private T value;
        private Node<T> next;
        private Node(T value, Node<T> next) {
            this.value = value;
            this.next = next;
        }
    }
    private Node<E> head;
    private int size;
    public int testRec(int i, int j) {
        return testRec(head, i, j);
    }
    private int testRec(Node<E> current, int i, int j) {
        if (j <= 0) {
            return 1;
        } else if (i <= 0) {
            return 1 + testRec(current.next, i, j-1);
        } else {
            return testRec(current.next, i-1, j-1);
        }
    }
    public void addLast(E elem) {
        if (elem == null) {
            throw new NullPointerException();
        }
        if (head == null) {
            head = new Node<E>(elem, null);
        } else {
            Node<E> p = head;
            while (p.next != null) {
                p = p.next;
            }
            p.next = new Node<E>(elem, null);
        }
        size++;
    }
    public String toString() {
        String res = "[";
        Node<E> cursor = head;
        while (cursor != null) {
            res = res + cursor.value;
            cursor = cursor.next;
            if(cursor != null) {
                res = res + ", " ;
            }
        }
        res = res + "]";
        return res;
    }

}
```

Given the implementation of the class **LinkedList** from the previous page, give the result that will be printed out on the console when executing the program below:

```
LinkedList<Integer> xs;
xs = new LinkedList<Integer>();
for (int i=0; i<10; i++) {
    xs.addLast(i);
}
System.out.println(xs);
System.out.println(xs.testRec(2,5));
```

Give your answer in the box below:

```
ANSWER:

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
4
```

Question 8 (10 marks)

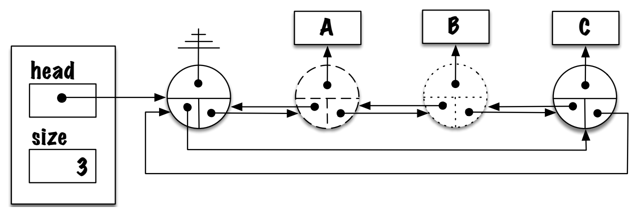
Implement the instance method **swapFirst** for the class **LinkedList** that has the following characteristics.

- An instance always starts off with a dummy node, which serves as a marker for the start of the list. The dummy node is never used to store data. The empty list consists of the dummy node only;
- In the implementation for this question, the nodes of the list are doubly linked;
- In this implementation, the list is circular, i.e. the reference **next** of the last node of the list is pointing at the dummy node, the reference **prev** of the dummy node is pointing at the last element of the list. In the empty list, the dummy node is the first and last node of the list, its references **prev** and **next** are pointing at the node itself;
- Since the last node is easily accessed, because it is always the previous node of the dummy node, the header of the list does not have (need) a tail pointer.

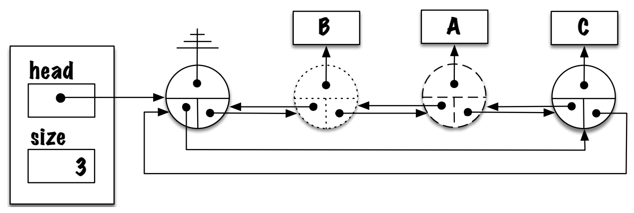
The method **swapFirst** exchanges the order of the first two nodes of the list.

- This question assesses your understanding of linked structures. Because of that, you cannot simply exchange the values. You need to exchange the order of the two objects of type **Node**.
- Likewise, you cannot use the methods of the class **LinkedList**. In particular, you cannot use the methods **add()** or **remove()**.
- The method throws the exception **IllegalStateException** if the size of the list is less than two (2).

Here is the memory diagram of the list before and after a call to the method **swapFirst**. We have used two different kinds of dotted lines to draw the first two nodes to identify the work to be done.



(before)



(after)

```
public class LinkedList<E> implements List<E> {

   private static class Node<T> {
      private T value;
      private Node<T> prev;
      private Node<T> next;
      private Node(T value, Node<T> prev, Node<T> next) {
         this.value = value;
         this.prev = prev;
         this.next = next;
      }
   }

   private Node<E> head;
   private int size;

   public void swapFirst() {

        if (size < 2) {
            throw new IllegalStateException();
        }

        Node<E> first, second, after;

        first = head.next;
        second = first.next;
        after = second.next;

        head.next = second;
        second.prev = head;

        second.next = first;
        first.prev = second;

        first.next = after;
        after.prev = first;

  }
}
```