L0201/L0202 Midterm 2002

CSC 148H
University of Toronto

Duration — 50 minutes

Student Number: |_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____

First Name: _____

TA: _____

---

*Do **not** turn this page until you have received the signal to start.*
Read this entire page or you'll miss the bonus question.

Midterm aids allowed:

- the Java API Reference booklet

Bonus question: if you write your student ID at the top of pages 2–6, you will get an extra mark.
Please write legibly.
If you run out of space on a question, use the back of the page.

1: _____/10

2: _____/10

3: _____/10

BONUS: _____/ 1

TOTAL: _____/30

Good Luck!

PLEASE HAND IN

Question 1. [10 MARKS]

For this question you will write a method which manipulates a Queue object. To do this you must use a stack to help you. Here are the Stack and Queue interfaces.

```
public interface Stack {
  /** Add o to my top */
  public void push(Object o);

  /** Remove and return my top element */
  public Object pop();

  /** Return true if I am empty and false otherwise */
  public boolean isEmpty();
}

public interface Queue {

  /** append o to me */
  public void enqueue(Object o);

  /** Remove and return my front element */
  public Object dequeue();

  /** Return the number of elements in me */
  public int size();
}
```

Assume further that we have a class `SomeStack` that implements the stack interface. You may also assume that neither your queue nor your stack ever become full.
Your job is to complete the method `reverse` which will reorder some of the elements in the queue. Here is a picture of the elements stored in `Queue q`:

```
head               tail
 |                   |
 a  b  c  d  e  f  g  h
```

Calling reverse(q,3,6) will result in the following situation:

```
head               tail
 |                   |
 a  b  c  g  f  e  d  h
```

Calling reverse(q,0,1) will further result in the following situation:

```
head               tail
 |                   |
 b  a  c  g  f  e  d  h
```

Part (a) [3 MARKS]

Write a method comment for `reverse` including any preconditions that are necessary.

Part (b) [7 MARKS]

Complete the method. You must make use of the `Stack temp` to do the reversal. Hint: It isn't helpful to instantiate a new object which implements Queue. Simply use the one in the method parameter.

```
public class QuestionOne {


    /** Considering the elements of q to be numbered 0 .. size() - 1,
      * with 0 being the head, we reverse the order in q of the elements
      * numbered start .. end.
      * Requires: 0 <= start < end < q.size().
      */
    public static void reverse(Queue q, int start, int end) {

        Stack temp = new SomeStack();

        for (int i=0; i<start; i++) {
            Object o = q.dequeue();
            q.enqueue(o);
        }
        for (int i=start;i<=end;i++) {
            temp.push(q.dequeue());
        }
        for (int i=start;i<=end;i++) {
            q.enqueue(temp.pop());
        }
        for (int i=end+1;i<q.size();i++){
            q.enqueue(q.dequeue());
        }
    }
}
```

Question 2. [10 MARKS]

Suppose we require that for `Stacks`, `pop()` throw an `EmptyStackException` (a type of `RuntimeException`) if the stack is empty. Then we can write `isEmpty()` by trying to pop an element.
If we make `Stack` an abstract class we can implement `isEmpty()`, letting subclasses implement `push()` and `pop()`.
Your job is to write the body for `isEmpty()`.

```
public abstract class Stack {

    /** Add o to my top */
    public abstract void push(Object o);

    /** Remove and return my top element.
     *  Throws EmptyStackException if there is none.
     */
    public abstract Object pop();

    /** Return true if I am empty and false otherwise */
    public boolean isEmpty() {

        try {
            push(pop());
            return false;
        } catch (EmptyStackException e) {
            return true;
        }

    }

}
```

Question 3. [10 MARKS]

```
public class M {
  public static int z;
  public M n;

  public void w(M e) {
    M p = this;
    while(p.n != this)
      p = p.n.i(p, e);
    p.n.i(p, e);
  }

  public M i(M p, M e) {
    e.n = p.n;     // Line 1
    p.n = e;
    return e.n;
  }

  public static void main(String[] s) {
    M a = new M();
    a.n = a; a.z = 99;
    M b = new N(55);       // Line 3
    a.n = b; b.n = a;
    a.w(new N(77));        // Line 5
    // ...
  }
}
```

```
public class N extends M {
  public int z;
  public M n;

  public N(int n) {
   z = n;
  }

  public M i(M p, M e) {
    return p.n;
  }
}
```

On the next page we have drawn a correct memory model for the point at which the above program reaches the beginning of line 3 in the main method.

Continue to trace this program until you first reach "Line 1" of method `i` in class `M`, which happens during the call to method `w` on "Line 5" of the main method. We recommend that you use the space below for any scratch work (we will NOT mark it), and then copy your final answer onto the picture on the next page.

Question 3. (CONTINUED)