

CSC 148H Midterm

Fall 2007

St. George Campus

Duration — 50 minutes

Student Number: _____

Family Name: _____

Given Name: _____

No Aids Allowed.

*Do **not** turn this page until you have received the signal to start.*

1: _____/10

2: _____/10

3: _____/10

TOTAL: _____/30

Good Luck!

PLEASE HAND IN

Question 1. [10 MARKS]

Write the class `RevQ`, which is to be a subclass of `CircularQueue`. The `RevQ.java` class definition should contain only one constructor and one method, as follows:

- a) A constructor `RevQ(int cap)`, where `cap` specifies the initial capacity of the queue.
- b) A public method `reverse()`, which reverses the FIFO ordering of the queue. That is, after calling `reverse()`, the element that used to be at the head of the queue is now at the tail. The element that used to be second is now second last, and so on. After reversal the queue is the same size as before. The method `reverse()` does not return anything. Finally, `reverse()` should throw a `CrazyUserException` when called on a queue of size less than or equal to one. An example is given at the bottom of this page.

You can assume `CrazyUserException` is as given below:

```
public class CrazyUserException extends Exception {
    public CrazyUserException() {}
    public CrazyUserException(String m) {super(m);}
}
```

Also, assume the classes `CircularQueue` and `ArrayStack`, along with the interfaces `Queue` and `Stack`, are already defined and are similar to the ones defined in the lectures. The **only non-private members** of these classes are described below.

Reminders: Use the following in this question:

- `Queue` has only the (public) methods `enqueue(Object o)`, `Object dequeue()`, `Object head()`, `int size()`, `int capacity()`.
- `Stack` has only the (public) methods `push(Object o)`, `Object pop()`, `int size()`, `int capacity()`.
- `CircularQueue` has only the constructor `CircularQueue(int capacity)`, and the public methods as specified by the interface `Queue`. Everything else is private.
- `ArrayStack` has only the constructor `ArrayStack(int capacity)`, and the public methods as specified by the interface `Stack`. Everything else is private.

An example of the use of this new class is as follows:

```
RevQ q = new RevQ(10);
q.enqueue("A");
q.enqueue("B");
q.enqueue("C");
q.head();    // returns "A"
q.reverse();
q.dequeue(); // returns "C"
q.dequeue(); // returns "B"
q.dequeue(); // returns "A"
q.size();    // returns 0
q.reverse(); // throws a CrazyUserException
```

Use the back of the last page for scratch work, and write your solution on the next page.

Question 1. (CONTINUED)

Question 2. [10 MARKS]

The `LinkedList` class implements the `Queue` interface (see the reminder in question 1). Assume the methods described in this interface have already been completed in `LinkedList.java`. Your job is to write the method `Object remove(int k)` below.

Here is the class definition for `ListNode`:

```
public class ListNode {
    public Object value;
    public ListNode link;
    public ListNode(Object o) {
        value = o;
    }
}
```

And here is the beginning of `LinkedList.java`

```
import java.util.NoSuchElementException;

public class LinkedList implements Queue {
    /** The node at the head of the queue, or
        null if size is zero. */
    private ListNode head;
    /** The node at the tail of the queue, or
        null if size is zero. */
    private ListNode tail;
    /** The number of items in the queue. */
    private int size;
    public LinkedList() {}

    ... Assume the method declarations for all the ...
    ... methods specified by Queue are here. ...

    /** Remove the k-th object in the queue. The head of the queue is
        at k = 0, the second item in the queue is at k=1, and so on.
        @param k the logical index within the queue of the item
            to be removed.
        @returns the data item just removed from the queue.
        @throws java.util.NoSuchElementException (a RuntimeException)
            when k is outside the range 0 <= k < size(). */

    public Object remove(int k) _____ {
        // Does anything HAVE to go in the underlined blank space above?
        // Complete this method (only). There is more space on the next page.
```

Question 2. (CONTINUED)

Question 3. [10 MARKS]

Draw the memory model for the situation where the 4th line of the main method is about to be executed. You do not need to draw `String` or `String[]` objects. There is more space on the last page.

```
public class Driver {
    public static void main(String[] args) {
1:   Item p = null;
2:   for (int k = 1; k < 4; k++)
3:       p = new Item(k);
4:   int k = p.x;
    }
}
```

```
public class Item {
    public static Item m;
    public int x;
    public Item p;
    public Item(int k) {
        x = k;
        rem(this);
    }
    public static void rem(Item q) {
        if (m == null)
            m = q;
        else {
            q.p = m;
            m = q;
        }
    }
}
```

Question 3. (CONTINUED)