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

Total pages = 7  Page 1  OVER...
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:

```java
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:

```java
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)

SOLUTION

public class RevQ extends CircularQueue {

    public RevQ(int cap) {
        super(cap);
    }

    public void reverse() throws CrazyUserException {
        if (size() <= 1)
            throw new CrazyUserException("Don’t flip.");

        Stack s = new ArrayStack(size());
        while (size() > 0) {
            s.push(dequeue());
        }
        while (s.size() > 0) {
            enqueue(s.pop());
        }
    }
}
Question 2.  [10 marks]

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

Here is the class definition for ListNode:

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

And here is the beginning of LinkedQueue.java

```java
import java.util.NoSuchElementException;

public class LinkedQueue 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 LinkedQueue() {} ...

    /** 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.
    }
```

... Assume the method declarations for all the ...  
... methods specified by Queue are here. ...
Question 2. (CONTINUED)

SOLUTION

public Object remove(int k) {
    if (k < 0 || k >= size())
        throw new NoSuchElementException("Index " + k + " out of range.");

    Object r;
    if (k == 0) { // Remove head
        r = head.value;
        head = head.link;
        size--;
        if (size == 0) // tail deleted!!
            tail = null;
    } else {

        ListNode prev = head;

        int cnt = 1;
        while (cnt < k) {
            prev = prev.link;
            cnt++;
        }
        // Assert: prev.link refers to k-th list node,
        // so it cannot be null.
        r = prev.link.value;

        if (prev.link == tail) { // tail changed
            tail = prev;
            prev.link = null;
        } else {
            prev.link = prev.link.link;
        }
        size--;
    }
    return r;
}
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) {
        Item p = null;
        for (int k = 1; k < 4; k++)
            p = new Item(k);
        int k = p.x;
    }
}

SOLUTION: See next page.
Question 3.

```
Driver
String[] args

void rem(Item)

Driver
Object
void main(String[])

Item p
null

Item p
x01

Item p
x02

Item p
x03
```

Total pages = 7
End of Solutions