

CSC 148H Midterm

Fall 2005

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.*

*Read this entire page or you'll miss the bonus question.*

---

# 1: \_\_\_\_\_/10

# 2: \_\_\_\_\_/10

Bonus question: if you legibly write your student ID at the top of each odd numbered page you will get an extra mark.

# 3: \_\_\_\_\_/10

BONUS: \_\_\_\_\_/ 1

TOTAL: \_\_\_\_\_/30

*Good Luck!*

PLEASE HAND IN

**Question 1.** [10 MARKS]

```

public class ArrayQueue
    implements Queue {

    public int size;
    public Object[] contents;

    public ArrayQueue(int n) {
        contents = new Object[n];
    }
    /** Precondition:
     *   The queue cannot be full. */
    public void enqueue(Object o) {
        contents[size++] = o;
    }
    /** Precondition:
     *   The queue cannot be empty. */
    public Object dequeue() {
        Object head = contents[0];
        for (int i = 0; i < size - 1; ++i) {
            contents[i] = contents[i + 1];
        }
        --size;
        return head;
    }
    /** Precondition:
     *   The queue cannot be empty. */
    public Object head() {
        return contents[0];
    }
    public int size() {
        return size;
    }
    public int capacity() {
        return contents.length;
    }
} // End of ArrayQueue.

```

The `ArrayQueue` class definition to the left is same as the one we discussed in class, except the instance variables `size` and `contents` are **public**. (While it is not a good idea in general to set these instance variables to have **public** accessibility, it is useful for the purposes of this exam.) Assume the interface `Queue` includes the methods headers (other than the constructor) of all the methods in `ArrayQueue`. Note that the first element in the queue (if any) is always stored at index 0 of the `contents` array, and the last element (if any) is at index `size-1`.

Write the Java code for a subclass of `ArrayQueue`, called `LeakyQ`, which has the following properties. It should have a **constructor which takes a single integer parameter** which specifies the (constant) capacity of the queue to be constructed. It should also have a **remove method which takes a single integer parameter** providing the index of a queue element to be removed. The **remove method** should return the `Object` that is removed from the queue. Any items which come after the removed item in the queue should be moved one space towards the front of the queue. If the parameter for **remove** is not between 0 and `size-1`, then the method should throw a `java.lang.IndexOutOfBoundsException`, which is a `RuntimeException`.

Your `LeakyQ` should not declare any instance variables. Moreover, it should only contain one constructor and one method, as described above.

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

```

LeakyQ q = new LeakyQ(10);
q.enqueue("A");
q.enqueue("B");
q.enqueue("C");
q.remove(1); // returns "B"
q.size();   // returns 2
q.head();   // returns "A"
q.remove(1); // returns "C"
q.size();   // returns 1
q.head();   // returns "A"

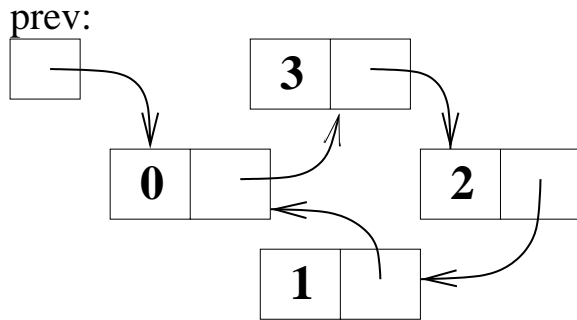
```

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 `LinkedRing` class provides a linked list of items where the last element in the list refers back to the first element. A sketch of a possible configuration for the `LinkedRing` is given below:



Write the `insert` method in the `LinkedRing` class according to the method comment below. For example, the state pictured to the left would arise from the following code:

```

LinkedRing lr = new LinkedRing();
lr.insert(new Integer(0));
lr.insert(new Integer(1));
lr.insert(new Integer(2));
lr.insert(new Integer(3));
  
```

Each item in the list is stored in a `ListNode`, which is defined as follows:

```

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

Finally, here is the beginning of the `LinkedRing` class definition:

```

public class LinkedRing {
    /** If prev == null then the list is empty,
     * otherwise, prev.link refers to the first item
     * on the list. */
    private ListNode prev;
    /** The number of items in the list. */
    private int size;

    public LinkedRing() {}

    /** Insert object o as the first element in the list, pushing all the other
     * items in the list one step further away from being the first item.
     * Postcondition: The link for last item in the list will refer to this
     * newly inserted item. */
    public void insert(Object o) {
        // 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 5th 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:   Point a = new Point(0, 0);
2:   Origin b = new Origin(a);
3:   Point c = b.get();
4:   a.x = 100;
5:   a.y = 50;
    }
}
```

```
public class Origin {
    private Point o;
    public Origin(Point p) { o = p; }
    public Point get() { return o; }
}
```

---

```
public class Point {
    public int x;
    public int y;
    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```

**Question 3.** (CONTINUED)