CSC 150H Midterm

Fall 2006

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: _____/ 5

2: _____/15

3: _____/10

TOTAL: _____/30

Good Luck! PLEASE HAND IN

Question 1. [5 MARKS]

```
public interface Queue {
  /** Add o to end of queue.
   *  Throws QueueFullException if
   *  the queue is currently full. */
  void enqueue(Object o);
  /** Remove and return the first
   *  element in the queue.
   *  Throws java.util.NoSuchElementException
   *  if the queue is currently empty. */
  Object dequeue();
  /** Return the number of items in
   *  the queue. */
  int size();
  /** Return the maximum number of elements
   *  this queue can hold. */
  int capacity();
}
```

```
public interface Stack {
  /** Add o to top of the stack.
   *  Throws StackFullException if
   *  the stack is currently full. */
  void push(Object o);
  /** Remove and return the top
   *  element in the stack.
   *  Throws java.util.NoSuchElementException
   *  if the stack is currently empty. */
  Object pop();
  /** Return the number of elements in
   *  the stack. */
  int size();
  /** Return the maximum number of elements
   *  this stack can hold. */
  int capacity();
}
```

Complete the method `reverse(Stack)` below. **Do not declare any additional variables.** (You may not need both of `t` and `q` declared below.) The method should take a given `Stack` and reverse the order of the elements. That is, the item at the top of the original stack will be at the bottom of the stack when the method returns, the item second from the top when the method is called will be second from the bottom when it returns, and so on.

The `Stack` and `Queue` interfaces are given above. Here all exceptions are subclasses of `RuntimeException`. You can assume that you have classes `MyStack` and `MyQueue` which implement the `Stack` and `Queue` interfaces, respectively. Assume both classes have constructors which accept one integer argument specifying the maximum capacity of the stack or queue (i.e., `MyStack(int capacity)` and `MyQueue(int capacity)`.

```
public class StackUtil {
  /** Reverse the order of the items in s.
   *  Requires: s is not null. */
  public static void reverse(Stack s) {
    Stack t = new MyStack(Math.max(1, s.size()));
    Queue q = new MyQueue(Math.max(1, s.size()));
```

SOLUTION

```
    while(s.size()>0)
      q.enqueue(s.pop());

    while(q.size()>0)
      s.push(q.dequeue());

  }
```

Question 2. [15 MARKS]

```
public class CircularQueue
               implements Queue {
  public Object[] contents;
  public int head;
  public int size;
  // Representation Invariant:
  //   0 <= head < capacity
  //   0 <= size <= capacity
  // If size > 0, the items in the queue
  // are stored in order from contents[head]
  // up to contents[(head+size-1)%capacity]
  // with wrap around, if necessary.
  // Here capacity = contents.length

  public CircularQueue(int capacity) {
    contents = new Object[capacity];
  }

  // The methods (with bodies) for the
  // methods specified in the Queue
  // interface on page 2 go here.
}
```

Note that the `CircularQueue` class given to the left is similar to the one discussed in the lectures, except the instance variables here are public. This is not a good idea in general, but is convenient for this exam question.

Write the definition for class `Deque` which should extend `CircularQueue`. `Deque` must not have any instance or static variables, and should only have a constructor and the following two methods. (**You do not need to complete the `CircularQueue` class definition.**)

```
/** Insert o at the front of the queue.
 *  Throws QueueFullException if the
 *  queue is already full. */
public void insertFront(Object o)

/** Remove and return the last element
 *  in the queue.
 *  Throws java.util.NoSuchElementException
 *  if the queue is initially empty. */
public Object removeBack()
```

SOLUTION:

```
import java.util.NoSuchElementException;

public class Deque extends CircularQueue {

  public Deque(int capacity) {
    super(capacity);
  }
  public void insertFront(Object o) {
    if (size == contents.length)
      throw new QueueFullException("Capacity "+
                                   contents.length);
    head = (head - 1 + contents.length) % contents.length;
    contents[head] = o;
    size++;
  }
  public Object removeBack() {
    if (size == 0)
      throw new NoSuchElementException("Queue is empty");
    int tail = (head + size - 1) % contents.length;
    size--;
    Object res = contents[tail];
    contents[tail] = null;
    return res;
  }
}
```

Question 3. [10 MARKS]

Draw the memory model for the situation where the main method below is executing, it has called the method `c.m(int)` on line number 4, and the last line of `m(int)` is about to be executed. You do not need to draw `String` or `String[]` objects. There is more space on the last page. This is not a trick question, the classes compile and the main method runs without an error.

```
  public class Driver {
    public static void main(String[] args) {
1:    int k = 10;
2:    B b = new B(null);
3:    B c = new B(b);
4:    k = c.m(k);
    }
  }
```

```
public class B {
  private static int v;
  private B b;
  private int w;
  public B(B b) {
    v++;
    this.b = b;
    w = v;
  }
  public int m(int j) {
    B p = this;
    int sum = j;
    while (p != null) {
      sum += p.w;
      p = p.b;
    }
    return sum;
  }
}
```

SOLUTION: See next page.

Question 3. (CONTINUED)