CSC 148H L0101 Midterm Fall 2003
Duration — 50 minutes
Aids allowed: none

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

Family Name: _____ **Given Name:** _____

---

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above,
and read the instructions below.) *Good Luck!*

This midterm consists of 3 questions on 4 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*
Comments are not required, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code.
Write your student number at the bottom of pages 2-4 of this test.
If you use any space for rough work, please indicate clearly what you want marked.

1: _____/12
2: _____/ 8
3: _____/10

TOTAL: _____/30

Question 1. [12 MARKS]

Complete the methods of class `SillyQueue`:

```
/**
  Uses two Queues of capacity 10 to implement
  a Queue of capacity 20.
 */
public class SillyQueue implements Queue {

  private Queue q1 = new CircularQueue(10);
  private Queue q2 = new CircularQueue(10);

  // Representation invariant:
  //   If there are at most 10 elements,
  //    q1 contains them in order and q2 is empty.
  //   If there are more than 10 elements,
  //    q1 contains the first 10 in order, and
  //    q2 contains the rest in order.

  public Object head() {

    return q1.head(); // 2 marks


  }

  public int size() {

    return q1.size() + q2.size(); // 3 marks


  }

  public void enqueue(Object o) {

    if (q1.size() < 10) { // 2 marks for logic
      q1.enqueue(o);      // 1 mark for reasonable enqueuing
    } else {
      q2.enqueue(o);
    }







  }
```

```
  public Object dequeue() {

    Object result = q1.dequeue(); // 1 mark for returning head()
    if (q2.size() > 0) {          // 1 mark for when to move element from q2 to q1
      q1.enqueue(q2.dequeue());   // 2 marks for moving element from q2 to q1
    }

  }
}
```

Question 2. [8 MARKS]

Consider the following class for nodes of a linked list:

```
public class Node {
  public Object value;
  public Node next;
}
```

Write the following method:

```
/* Returns the linked list composed of only the mth to nth *nodes*
   of the linked list referred to by 'list'.
   (This will usually change the original linked list as well).
   For example, subList(list, 1, 4) returns the first four nodes of list.
   Requires: list refers to a linked list of at least n nodes, 1 <= m <= n. */
public static Node subList(Node list, int m, int n) {

  for (int i = 1; i < m; ++i) { // 2 marks
    list = list.next;
  }

  Node tail = list; // 1 mark
  for (int i = m; i < n; ++i) { // 2 marks
    tail = tail.next;
  }
  tail.next = null; // 2 marks

  return list; // 1 mark









}
```

Question 3. [10 MARKS]

```
public class C {                  |          public class D extends C {
  public static int i;            |            public int i;
  public void p(int j) {          |
    m(j);                         |
  }                               |
  public void m(int j) {          |            public void m(int j) {
    i = i + j;                    |              i = i + j;
    s();                          |              s();
  }                               |            }
  public static void s() {        |          }
    i = i + 1000;                 |
  }                               |
}                                 |
public class M {
  public static void main(String[] a) {
    C c = new C();
    c.p(1);
    D d1 = new D();
    d1.m(10);
    C d2 = new D();
    d2.p(100);
  }
}
```

Draw the memory model for this program just after the 3rd time that `s()` returns (but before the method that calls `s()` returns).