University of Toronto
CSC148 – Introduction to Computer Science, Summer 2003

Mid Term Test – Section L0101

Duration: 50 minutes
Aids allowed: 1 8.5" x 11" piece of paper with information written on one or both sides.

Make sure that your examination has 6 pages (including this one). Write your answers in the spaces provided. Write legibly. You may use page 6 for rough work (tear it off, if you like). If you require more space to answer a question, write on the back of the previous page, and indicate in the answer space where your answer is.

Personal Information:

Surname:	
Given name(s):	
Student #:	
Circle section of registration:	L0101 or L5101

Please note that if you write the midterm in pencil, you will ***not*** be allowed to submit a remark request.

Do not write anything on this page below this line:

1.	/ 12
2.	/ 12
3.	/ 6
Bonus.	/ 0
Total:	/ 30

Question 1: Quick Concept Questions

[3 sub-questions, 4 marks each]

a. Briefly explain the difference between a `null` String (eg: `String s = null;`) and an empty String (eg: `String s = "";`). Include a description of the differences seen in the Object Space of computer memory.

A null String stores the value "null" in the memory space for the reference, and contains no reference to anything in Object Space. An empty string is a reference to an instance of the String class in Object Space that is storing the value "".

3 marks for stating empty is an instance in the object space, null is not
1 mark for stating that s is a reference to that empty string object or contains "null"

b. Below is the skeleton of a try/catch block:

```
try {
   // block 1: code that throws an exception
} catch (ExceptionClass1 e) {
   // block 2: code that deals with this type of exception
} catch (ExceptionClass2 e) {
   // block 3: code that deals with this type of exception
}
```

An exception thrown in block 1 might match either ExcpetionClass1 or ExceptionClass2. Under what circumstance would it match both? Which block of code (2 or 3) would run if this were to happen?

If ExceptionClass1 extends ExceptionClass2 (or vice versa), and the thrown exception is the sub-class, then it matches both. Block 2 would be the one to run.

3 marks for saying one of EC1 or EC2 extends the other (or exists in same path in inheritance hierarchy tree)
1 mark for saying that more specific (sub-class) must be thrown

c. What special name have we given to trees with a branching factor of 1?

Singly-Linked List

4 marks for this or simply "Linked List"

Question 2: Linked List

[12 marks]

Here is a node class:

```
class IntNode {
   int data;
   IntNode next;

   IntNode(int data, IntNode next) {
      this.data = data;
      this.next = next;
   }
}
```

Complete the method body for "getMax" below. ***Be sure to include all internal comments.***

```
/* Returns the maximum value stored in the list rooted at head.
 *
 * Requires: the list contains at least 1 element. */

public static int getMax(IntNode head) {

      // Algorithm: traverse the linked-list, examining each
      // element, keeping a reference to the max.

      IntNode max = head; // assume the first element is the max
      IntNode temp = head.next;

      while (temp != null) {

            // if we've found a new max
            if (temp.data > max.data) {
                  // store it
                  max = temp;
            }
            temp = temp.next;
      }

      return max.data;
```

1 mark for algorithm comment at top of method
1 mark for comments throughout method explaining **correct** algorithm at a high level
4 marks for correctly traversing the list, finding the max, and returning it.

} ***// did you remember to return something, and include comments?***

Complete the method body for "insertBefore" below (using the IntNode class from the previous page). ***Be sure to include all internal comments.***

```
/* Inserts "n" into the list rooted at "head" before the node
 * containing the value "i" in the list.
 * Returns the head of the list containing n.
 *
 * Requires: there is a node in the list rooted at head
 * containing i. */

public static IntNode insertBefore(IntNode head, int i, int n) {

   // algorithm: traverse the list, looking one node ahead, and
   // stop
   // on the node before i.  Insert the new node ahead of that
   // node.

   // if we have to insert before the head
   if (head.data == i) {
      // store the head value
      head = new IntNode(n,head);
   } else { // head does not contain i, look elsewhere
      IntNode temp = head;
      while (temp.next.data != i) { temp = temp.next; }
      temp.next = new IntNode(n,temp.next);
   }

   return head;
```

1 mark for algorithm comment at top of method
1 mark for comments throughout method explaining **correct** algorithm at a high level
2 marks for identifying and dealing correctly with head.data == i case
2 marks for identifying and dealing correctly with more general case (head.data != i)

```
} // did you remember to return something, and include comments?
```

Question 3: Recursion

[6 marks]

Here is a recursive method:

```
/* Finds and returns the sum of the series [0,1,…,n]
 * Requires: n>= 0
 */

int sumLess(int n) {

   if (n == 0) {
      return 0;
   } else {
      // A: return sumLess(n);
      // B: return n + sumLess(n);
      // C: return n + sumLess(--n);
      // D: return n + sumLess(n--);
   }
}
```

In order for sumLess to work as specified, which line of code should be uncommented? Circle the correct answer:

The answer is C 4 marks for circling the correct answer

A ***B*** ***C*** ***D***

Bonus question

[+3 marks]

Would the line:

```
return (n--) + sumLess(n);
```

work in place of the commented lines above? Circle one of:

YES ***NO***

Explain your answer:
Because the value of "n--" is "n", but the effect is to decrease n by one, the above line is effectively equivalent to: return (n) + sumLess(n-1);

The recursive call to sumLess will return the sum of [0,1,…,n-1]. We need only add "n" to it to yield the sum of [0,1,…,n].

0 marks for circling "yes"
3 marks for correctling explaining that the above line is equivalent to "return n + sumLess(n-1)"

This page intentionally left blank. Tear it out and use it for scratch paper.