University of Toronto
CSC148 – Introduction to Computer Science, Summer 2004 // Daniel Wigdor

Mid Term Test

Duration: 50 minutes
Aids allowed: none

Make sure that your examination has 8 pages (including this one). Write your answers in the spaces provided. Write legibly. You may use page 8 for rough work (tear it off, if you like). If you require more space to answer a question, write them on the back of the previous page, and indicate in the answer space where your answer is.

Personal Information:

Surname:	
Given name(s):	
Student #:	
Lab Section (circle exactly one)	• **Tuesday 4-6pm @ 2200 w/ Marcel Leica** • **Tuesday 4-6pm @ 2240 w/ Edward Xia** • **Wednesday 11-1pm @ 2240 w/ Yun Niu** • **Wednesday 12-2pm @ 2200 w/ Yuntian Fan**

Please note that if you write the midterm in pencil, your midterm will ***not*** be re-graded under any circumstances.

Do not write anything on this page below this line:

1.	/ 8 (max 8 / 8)
2.	/ 8 (max 8 / 8)
3.	/ 8 (max 8 / 8)
Bonus.	/ 3 (max 8 / 3)
Total:	/ 27 (max 32 /27)

Question 1: Design by Contract

[8 marks]

Answer each of the following in the space provided. Be concise!

1. It is generally the case that method contracts deal only with what a method does, rather than how it does it. What is the one exception to this rule?
[3 marks]

   Important performance facts should be included in the external comment.

2. Below, find the code for the "fun" method. Provide a complete method contract for this method, following all the rules for proper method contracts. To save time, don't bother with comment syntax – just write the English text of the comment. (Don't worry – this isn't a trick question! The code compiles and performs a useful task. You may assume that a simple `StopException` class exists).
[5 marks]

```
/*
 * Returns the index of the first occurrence of o in a, or -1
 * if a does not contain o
 * Requires: a != null, no element of a is null
 */


public static int fun(Object a[], Object o) {
   int i=0;
   try {
      for(i=0;;i++) {
         if (a[i].equals(o)) {
            throw new StopException();
         }
      }
   } catch (ArrayIndexOutOfBoundsException e) {
      return -1;
   } catch (StopException e) {
      return i;
   }
}
```

Question 2: Linked List

[8 marks]

To the right, find a node class:

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

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

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

```
/* Insert n into the linked list rooted at head, maintaining the
 * non-increasing order of the list, and return the (possibly
 * new) head of the list.
 * eg: if the list before insertion is 41->30->30->13, after
 * inserting 14 the list would be 41->30->30->14->13.
 *
 * Requires:
 *    The list rooted at head is a null terminated, singly-linked
 *    list of IntNodes linked in non-increasing order.
 */

public static IntNode insert(IntNode head, int n) {


     if (head == null || head.data <= n) {
          return new IntNode(head,n);
     } else {
          IntNode temp = head;
          // assert: temp!=null
          while (temp.next!=null && temp.next.data > n) {
               temp=temp.next;
          }
          temp.next = new IntNode(temp.next,n);
          return head;
     }

} // did you remember to include comments?
```

Question 3: Memory Model

[8 marks]

Here is a class, which includes a `main` *method. On the next page, draw the complete memory model for this class, starting with the execution of the* `main` *method. Stop your drawing at the comment labeled "stop here". It is alright to cross things out and write new values into boxes – make sure you allow lots of room to make updates to your diagram as the state of memory changes.*

Note: *Only the final product will be marked – no marks will be assessed by examining old values that you have changed. Also, for your convenience, this code appears again on page 8, which you may remove from the test booklet.*

```
public class FunTimes {

      public static int silly = 0;
      public String x = null;

      public void foo(String x) {

         /*1*/ x = "";
         /*2*/ silly++;

         /* stop drawing the first time you reach this. The line
         number in your diagram should be “2” */

         /*3*/ int i = 0;

      }
      public static void main(String[] args) {

         /*1*/ FunTimes f1 = new FunTimes();
         /*2*/ FunTimes f2 = new FunTimes();

         /*3*/ String s = "hi";

         /*4*/ f1.foo(s);
         /*5*/ f2.foo(s); //drawing will not reach this line
      }
}
```

Static Space

Object Space

Stack

Question 4: Bonus Question

[3 marks]

This question is optional – you may choose not to complete it, and receive a grade of 3/3. If you choose to complete it, your response will be graded out of 8, and your score will be added to your total grade for the test. Of course, it is possible to get less than 3/8, so ask us to grade it only if you are fairly confident of your response.

Please ***check*** *one of the following:*

	Do not grade the answer on page 7; I am happy with a 3 mark bonus.
	Do grade the answer on page 7; I realize I could get as low as 0 / 3, and as high as 8 / 3.

On the next page, find the beginning of a method, `reverseList`. Complete the method, **without declaring any more variables**. Below, find the Stack interface and the code for the `ListNode` class used by `reverseList`:

```
public class ListNode {

   public Comparable c;
   public ListNode next = null;

}
```

```
public interface Stack {

   /*
    * Place o at the top of the stack
    */
   public void push(Object o);

   /*
    * Remove and return the object at the top of
    * the stack
    * Requires: size()!=0
    */
   public Object pop();

   /*
    * Get the size of this Stack
    */
   public int size();

}
```

```
/* Reverses the order of the nodes in the linked-list
 * pointed to by r, and returns a reference to the new head
 * of the list.  The list is reversed in place – no
 * new ListNode objects will be created.
 *
 * Requires: r != null
 */
public static ListNode reverseList(ListNode r) {

      Stack s = new VectorStack();
      ListNode t = null;

      while (r!=null) {
            s.push(r);
            r = r.next;
      }
      r = (ListNode)s.pop();
      t = r;
      while (s.size()!= 0) {
            r.next = (ListNode)s.pop();
            r=r.next;
      }
      r.next = null;
      return t;

}
```

This page intentionally left blank. You may tear it out and use it for scratch paper.

Code from Question 2:

```
public class FunTimes {

      public static int silly = 0;
      public String x = null;

      public void foo(String x) {

         /*1*/ x = "";
         /*2*/ silly++;

         /* stop drawing the first time you reach this. The line
         number in your diagram should be “2” */

         /*3*/ int i = 0;

      }
      public static void main(String[] args) {

         /*1*/ FunTimes f1 = new FunTimes();
         /*2*/ FunTimes f2 = new FunTimes();

         /*3*/ String s = "hi";

         /*4*/ f1.foo(s);
         /*5*/ f2.foo(s); //drawing will not reach this line
      }
}
```

NOTHING ON THIS PAGE WILL BE GRADED