University of Toronto

csc148— Introduction to Computer Science Fall 1999

Midterm Test

Duration: 50 minutes

Aids allowed: None

Make sure that your examination booklet has 6 pages (not including this one). Write your answers in the spaces provided. Write legibly.

Surname: ______________________ **First name:** ______________________

Student #: ______________________

Tutor (circle one): Kaytek Przbylski Petter Wiberg Jade Rubick

1.	________	/ 8
2.	________	/ 7
3.	________	/ 6
4.	________	/ 10
Total	________	/ 31

Question 1 [8 marks in total]

Consider the `removeDuplicates` method shown below. It uses the following node class:

```
class Node {
    public Object contents;      public Node link;
    // Constructor.
    public Node (Object o) {
        contents = o;
        link = null;
    }
}

// Removes all but the first occurrence of ‘o’ in the linked list referred
// to by ‘front’.  Uses ‘.equals’ to check for equality.

// Preconditions:

public static void removeDuplicates (Node front, Object o) {

     Node temp = front;
     while (temp != null && !temp.contents.equals(o)) {
          temp = temp.link;
     }
     if (temp != null) {

          // Assertion:  temp is

          Node temp2 = temp;
          while (temp2 != null && temp2.contents.equals(o)) {
               temp2 = temp2.link;
          }

          // Assertion:  temp2 is

          temp.link = temp2;
     }
}
```

(a) [2 marks]
Under some circumstances, this method does not work as described in the comments. Add an appropriate precondition to the method comments.

```
Answer:
    // Preconditions: Elements of the list that are ".equals"
    //                occur consecutively.

A way to say the same thing more precisely:
    // Preconditions: There does not exist i <= j <= k such that
    //                the ith element of the list .equals the kth element,
    //                but does not .equal the jth.
```

CONTINUED

```
Another answer that isn't as good:
    // Preconditions: The list is sorted.

This would make the method work, but is not *required* to make the method work.
It is sufficient that the ".equals" elements occur consecutively.

Things that are *not* preconditions:
    front is not null
    o is not null
```

(b) [4 marks]
Where marked in the comments, write an appropriate assertion about `temp` or `temp2`, as indicated.

```
First assertion:
    // Assertion: temp refers to the first node in the list that .equals o.

It would be wrong to allow for the possibility that temp is null, since
we know that can't be the case at this point in the code.

Second assertion:
    // Assertion: temp2 refers to the first node after temp's node
    //            that doesn't .equal o.
```

(c) [2 marks]
`temp` and `temp2` are terrible variable names. Rename them.

New name for `temp`:
New name for `temp2`:

```
There are many good answers to this question.  The challenge is to express
essentially what the assertions say without having a ridiculously long name.
One good solution:
       firstMatch and nextNonMatch
```

CONTINUED

Question 2 [7 marks in total]

Complete the `insert` and `appendList` methods below. (They use the same `Node` class as question 1.) Note that `appendList` should not construct any new nodes; it should simply hook together the existing lists.

```
    // Inserts ‘o’ at the front of ‘list’ and returns the new front node.
    // Preconditions: none.
    public static Node insert (Node list, Object o) {


         Node temp = new Node(o);
         temp.link = list;
         return temp;




    }

    // Appends ‘list2’ to the end of the ‘list1’ and returns a reference to
    // its new front node.
    // Example: If ‘list1’ contains 1, 2, 3 (in that order)
    //         and ‘list2’ contains 7, 8, 9, 10 (in that order)
    // returns a reference to a list containing 1, 2, 3, 7, 8, 9, 10.

    // Preconditions: none.
    public static Node appendList (Node list1, Node list2) {



         if (list1 == null)
              return list2;
         else {
              Node temp = list1;
              while (temp.link != null)
                   temp = temp.link;
              temp.link = list2;
              return list1;
         }



    }
    // Other methods would go here.
}
```

CONTINUED

Question 3 [6 marks in total]

Consider the problem of representing a set of clients of a law firm. Below is an outline for code to represent the set.

Rewrite the outline to improve its design. You may change anything or add anything, but do not add any more detail than is already here. (For example, do not add new data or write any method bodies.) Make your changes directly on the code below.

```
class LinkedClientSet {

     // First client in a linked list.
     private Client firstClient;


     // Adds client 'c' to this set.
     public void add (Client c) {   // Details omitted.  }


     // Removes and returns a client from the set.
     // Precondition: the set is not empty.
     public Client remove () {   // Details omitted.  }


     public boolean isEmpty () {   // Details omitted.  }
}


class Client {
     public String name;      public String address;
     // Other client information would go here.

     // The next client in a linked list.
     public Client next;
}
```

CONTINUED

```
There are at least 3 major problems with the code as it is:
   - The Client class mixes together things to do with clients and things to
     do with nodes.  This is exactly analogous to one of the problems in the
     "bad design" discussion on the course web page.
   - The LinkedClientSet class could easily be generalized to handle any sort
     of object, and thus be useful in many other programs.
   - There should be an interface that the LinkedClientSet class (better named
     just LinkedSet) implements.

I may also be a good idea to make the instance variables in Client
private.  I have said in class that it is acceptable to make the instance
variables of a Node class public, because it is not a fully-fledged class
(it has no real methods).  Such a class is defining something more like
a "record" or "struct" in another programming language than an actual object.
By this same argument, the Client class could also have public instance
variables.

Here is a revised outline that fixes these problems:

interface Set {

     public void add (Object c);
     public Object remove ();
     public boolean isEmpty ();
}

class LinkedSet implements Set {

     // First node in a linked list of Objects.
     private Object first;

     // Adds Object 'c' to this set.
     public void add (Object c) {   // Details omitted.  }

     // Removes and returns an Object from the set.
     // Precondition: the set is not empty.
     public Object remove () {   // Details omitted.  }

     public boolean isEmpty () {   // Details omitted.  }
}

class Client {
     public String name;      public String address;
     // Other client information would go here.
}

class Node {
     public Object contents;

     // The next Node in a linked list.
     public Node next;
}
```

CONTINUED

Question 4 [10 marks in total]

(a) [6 marks]
Fill in the boxes to show the conclusion that is valid at each point in the following proof:

Assume P is true
  Assume Q is false
    Assume R is false
      ⋮
      Therefore, some contradiction holds.
    $\boxed{R \text{ is true}}$
  $\boxed{Q \text{ is false implies } R \text{ is true}}$
$\boxed{P \text{ is true implies that } (Q \text{ is false implies } R \text{ is true})}$

(b) [2 marks]
The following structure for a proof by induction jumps up by 2 in the induction step. Fill in the box in a way that makes the proof valid.

BASE CASES: Prove $S(4)$ is true and $S(5)$ is true.

Let $k \geq \boxed{4}$ be an arbitrary integer.

INDUCTION HYPOTHESIS: Assume $S(k)$ is true.

INDUCTION STEP: Prove $S(k+2)$ is true.

CONCLUSION: $S(n)$ is true for all $k \geq 4$

(c) [2 marks]
The induction proof structure above has two base cases. If we remove the second base case, what value should go in the box so that the proof will be valid? Circle one answer.

0
1
4
5
6
[✓ no value will make it valid]

END OF TEST