University of Toronto
csc148—Introduction to Computer Science, Fall 1997

First Term Test — Horton's evening section

Solutions!

Duration: 50 minutes

Aids allowed: None

Make sure that your examination booklet has 6 pages (including this one). Write your answers in the spaces provided. Write legibly.

Family Name: _____ **Lecturer:** _____

First Name: _____ **Tutor:** _____

Student # : _____

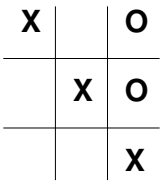
1. _____ / 7
2. _____ / 11
3. _____ / 9
4. _____ / 8

Total _____ / 35

Question 1

[7 marks in total]

Tic-tac-toe is a very simple game, played on a 3-by-3 grid. There are two players, named player X and player O. They take turns writing their symbol, X or O, into an empty spot on the grid. A player wins, and the game ends, if their symbol occurs in 3 consecutive spots, either vertically, horizontally, or diagonally. Below is an example game where player X has won.



It is also possible that neither player wins.

Define a simple ADT for a tic-tac-toe grid.

```
A tic-tac-toe board consists of:
 - a 3-by-3 square of entries, each of which is either marked
       X or O, or is not marked.

The operations one can perform on a tic-tac-toe board are:
 - mark spot: marks the given spot with the given mark, which must
       be X or O.  The square must not already be marked.
 - initialize board: returns a board with all spots unmarked.
 - marked: returns true if the given spot is already marked, and
       false otherwise.
 - X won: returns true iff
       there are 3 X's in sequence in
       some row, column, or on some diagonal.
 - O won: returns true iff
       there are 3 O's in sequence in
       some row, column, or on some diagonal.
 - tie: returns true iff (neither X won nor O won, but every spot is marked).
```

Question 2

[11 marks in total]

This question uses the `IntNode` class, given here:

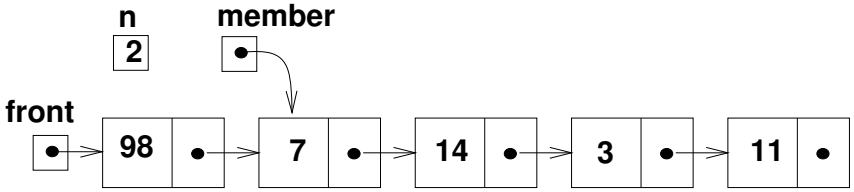
```
class IntNode {
   public int num;
   public IntNode next;

   // Constructor
   public IntNode(int n) {
      num = n;
      next = null;
   }
}
```

We need to write a body for the following Java method:

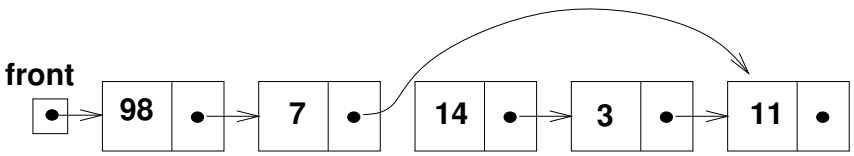
```
public static IntNode removeNodes (IntNode front, IntNode member, int n) {
```

This method takes `front`, a reference to the front node in a linked list; `member`, a reference to some node within that linked list (possibly the front one, possible not); and an integer `n`. The method must delete from the linked list `n` nodes after the node referenced by `member`, and return a reference to the front of the list.

For example, if `removeNode` were called with the following linked list and parameter values:



then it should do the following to the list, and return a reference to the front node:



(a) [2 marks]
We chose to have the method return a reference to the front of the revised list. Was this necessary? Circle the appropriate answer:

Yes **No**

Briefly explain your answer.
Because "member" has to refer to a node in the linked list, the earliest node it could refer to is the first one. Even then, any nodes to be deleted are deleted *after*

that one. So there is no way that the "front" reference will have to change. Returning it (unchanged) is thus unnecessary.

If the front could possibly be changed by the method, we'd have to return its (possibly new) value. The calling code would then have to store it into the appropriate variable. We cannot accomplish the same effect simply through the parameters because Java parameters are pass-by-value. (So any change the the formal parameter will not change the actual parameter.)

(b) [7 marks]
Complete the method. You may assume that all of the preconditions listed below are true.

```
// Preconditions: front must not be null;
//                member must not be null;
//                n >= 0; and
//                there are indeed n nodes after the node referenced by member.
public static IntNode removeNodes (IntNode front, IntNode member, int n) {



          IntNode temp = member;
          for (int i=1; i <= n+1; i++)
                  temp = temp.next;

          member.next = temp;
          return front;
   }
```

(c) [2 marks]
How does Java know that the method labelled a constructor (in our comments) is a constructor?

because (1) it has the same name as the class, and (2) it has no return type -- not even "void". All constructors must satisfy both of these requirements.

Question 3

[9 marks]

The code in this question uses the same `IntNode` class as used in Question 2. For each code segment below, circle the right answer to indicate whether it will crash or not. If it crashes, explain why. If it doesn't crash, show what the output would be.

```
IntNode n1 = new IntNode(3);
IntNode n2 = new IntNode(3);
IntNode n3 = n1;
if (n1 == n2)
   System.out.println("alpha");
if (n1 == n3)
   System.out.println("beta");
```

will crash because: | will run and give the following output:

```
beta
```

```
IntNode[] list;
list = new IntNode[15];
System.out.print("The contents of list: ");
for (int i=0; i<15; i++)
    System.out.print(list[i].num);
```

will crash because: | will run and give the following output:

Because we have constructed the array, but its elements are simply null; we haven't constructed any IntNodes for the array elements to refer to. So the first time we say "list[i].num", we are trying to dereference a null pointer. Crash.

```
IntNode[] anotherList = new IntNode[5];
anotherList[3] = new IntNode(99);
IntNode temp = new IntNode(101);
temp.next = anotherList[3];
anotherList[3] = temp;
System.out.println(anotherList[3].num);
```

will crash because: | will run and give the following output:

```
101
```

Question 4

[8 marks in total]

Suppose you are writing a program to keep track of inventory in a warehouse. The following class might be useful:

```
class InventoryItem {
    private int itemID;
    private int inStock;

    public InventoryItem(int identifier) {
         itemID = identifier;
         inStock = 0;
    }
    public void recordArrival (int numUnits) {
         inStock = inStock + numUnits;
    }
    public void recordShipment (int numUnits) {
         inStock = inStock - numUnits;
    }
    public int numberOfUnits () {
         return inStock;
    }
}
```

(a) [6 marks]
Write a code fragment that creates an array of 1000 inventory items, and records the fact that 50 units of each inventory item have arrived in the warehouse. The ID numbers of the items should correspond to their array positions. That is, inventory item 52 should be in array position 52.

```
InventoryItem[] A = new InventoryItem[1000];
for (int i=0; i< 1000; i++) {
        A[i] = new InventoryItem(i);
        A[i].recordArrival(50);
}
```

(b) [2 marks]
What does it mean to say that the `inStock` counter is private?

It means that no class other than InventoryItem can have access to that variable.

Why should it be private?

Because it is an implementation detail of the class. Client code doesn't *need* to know about it, and should not become dependent upon it -- it could change later if we decide to reimplement the class. Also, we don't want client code to, intentionally or not, harm it by giving it an inappropriate value.