University of Toronto
**csc148—Introduction to Computer Science, Spring 1998**

# Second Test — Horton's section

**Duration:** 50 minutes

**Aids allowed:** None

Make sure that your examination booklet has 5 pages (including this one). Write your answers in the spaces provided. Write legibly.

**Family Name:** _____

**First Name:** _____

**Student # :** _____

**Tutor (circle one):**

Mary Ellen Foster  David Suydam  Nitin Ramdenee  Neil Gower  Michael Neff

1. _____ / 9
2. _____ / 18
3. _____ / 8

Total _____ / 35

## Question 1

[9 marks in total]

Consider the following fragment of Java code:

```
int best = list[i];
i++;
while (list[i] != -1) {
    if (list[i] < best)
          best = list[i];
    i++;
}
```

*(a)* Give appropriate preconditions for this code.

*(b)* Give appropriate postconditions for this code.

First, give the most important postcondition — to do with the purpose of the code:

Now give any other postconditions:

*(c)* Give appropriate loop invariants for the loop.

First, give a loop invariant about the value of `best`:

Now give a loop invariant about the value of `i`:

## Question 2

[18 marks in total]

**Hint:** Each part of this question is independent of the others, and so can be answered even if you have not solved the others.

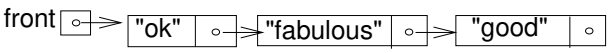
Consider the recursive Java method show below. It uses the given `Node` class.

```
class Node {
    public String data;
    public Node next;
}

private static int blah (Node p) {
    int ans;
    if (p == null) {
        System.out.println ("bottom out");
        ans = -1;
    } else {
        System.out.println ("p has: " + p.data);
        int rest = blah(p.next);
        if (p.data.length() > rest)
              ans = p.data.length();
        else
              ans = rest;
    }
    System.out.println ("ans is:" + ans);
    return ans;
 }
```

*(a)* [4 marks]

Assume the following linked list has been created:




Show the output that would result from the call `Blah(front)`, which runs without crashing.

*(b)* [4 marks]
Complete the specification of correct behaviour for the method, begun below.
Let $S(n)$ represent the following statement:

> "If `blah` is called with a reference to a linked list ...

*(c)* [4 marks]
Say we want to prove that $S(n)$ is true for all $n \geq 0$. Fill in the blanks so that the following proof structure will be valid, in other words, so that the conclusion will be valid (assuming that the sub-proofs in the base case and induction step are properly completed).

BASE CASE(S): Prove that _____

Let $k$ be an arbitrary integer _____

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

INDUCTION STEP: Prove that $S(k+1)$ is true.

INDUCTION CONCLUSION: $S(n)$ is true for all $n \geq 0$.

*(d)* [4 marks]
Fill in the blanks so that the following alternative proof structure will be valid, in other words, so that the conclusion will be valid (assuming that the sub-proofs in the base case and induction step are properly completed). The difference here is in the goal of the induction step.

BASE CASE(S): Prove that _____

Let $k$ be an arbitrary integer _____

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

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

INDUCTION CONCLUSION: $S(n)$ is true for all $n \geq 0$.

*(e)* [2 marks; no marks awarded without correct explanation]
Which proof structure is a better choice: that in (c) above, with $S(k+1)$ the goal of the induction step, or that in (d) above, with $S(k+2)$ the goal of the induction step? Circle the best answer.

(c): with $S(k+1)$ (d): with $S(k+2)$

Explain:

## Question 3

[8 marks in total]

The `TreeNode` class given below can be used for a binary tree of words.

```
class TreeNode {
    public String word;
    public TreeNode left, right;
}
```

Complete the method `makeString`, begun below. It must return a String containing all the words stored in the tree, in pre-order. For example, if we had a small tree with "hi" at the root, "you" in its left child, and "me" in its right child, the method would return the String "hiyoume".

```
public static String makeString (TreeNode root) {
```