University of Toronto
csc148F—Introduction to Computing, Fall 1996

Second Term Test — Horton's 6:00 section

Duration: 50 minutes

Aids allowed: None

Make sure that your examination booklet has 4 pages (including this one). Write your answers in the spaces provided. Write legibly.

Family Name: SOLUTIONS

Lecturer: ______________________

First Name: ______________________

Tutor: ______________________

Student # : ______________________

1.	________	/ 8
2.	________	/ 3
3.	________	/ 10
4.	________	/ 14
Total	________	/ 35

Question 1

[8 marks]

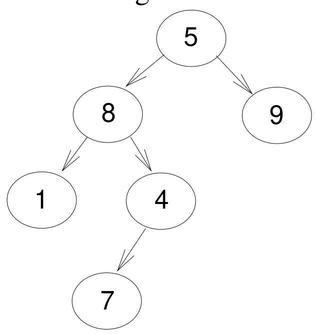
Complete the following OOT function, whose purpose is to return the maximum value in a non-empty linked list of integers.

```
type listNode :
    record
        num : int
        link : ^listNode
    end record

% Precondition: front is not nil
function biggest (front : ^listNode) : int

     var temp : ^listNode := front
     var answer : int := front -> num
     loop
         temp := temp -> link
         exit when temp = nil
         if temp -> num > answer then
             answer := temp -> num
         end if
     end loop

     result answer

 end biggest
```

Question 2

[3 marks]

Consider a subprogram whose purpose is to print the nodes of a tree in post order. In what order would it print the nodes of the following tree:



(Recall that in a post-order traversal of a tree, each node is processed after its children have been processed in order.)

ORDER:

1 7 4 8 9 5

Question 3

[14 marks in total]

Consider the following OOT function:

```
function power (n, p : int) : int

    % Preconditions: p >= 0  and  n has a value
    var answer : int := 1
    var count : int := 0
    loop
        % Loop invariant: answer = n^count.
        % (That is, answer = n to the power of count.)
        exit when count = p
        answer := answer * n
        count := count + 1
    end loop
    % Postconditions: answer = n^p

    result answer
end power
```

(a) [2 marks]
Trace a call to power (2, 4), and fill out the table below to show the sequence of values taken by the variables.

n	p	answer	count
2	4	1	0
2	4	2	1
2	4	4	2
2	4	8	3
2	4	16	4

(b) [3 marks]
To prove termination of a loop, we defined an integer-valued quantity that decreases on each iteration, yet can never be less than zero. (For example, last-first, in our binary search proof.) Give an appropriate such quantity to use for proving termination of the loop in function power.

```
p - count
```

(c) [1 marks]
Write a loop invariant that would help us to prove termination of the loop. Hint: Your answer here will be based heavily on your answer to part (c).

```
(p - count) >= 0

Another way to say it:
     count <= p
Another way, with more information:
     0 <= count <= p
```

(d) [8 marks]
Prove partial correctness of this function. That is, prove that:

> If the preconditions are true, the code is executed, and the code terminates, then the postconditions will be true.

You may use the loop invariant, assuming that it has been proven. Hint: This proof will take just a few lines.

```
A very detailed solution:

     Assume that the preconditions are true, the code is executed,
     and the code terminates.

         - answer = n^count is a loop invariant.
           By definition of loop invariant, it must be true at the
           beginning and ending of every iteration.
         - So it must be true at the end of the last iteration.
         - It must still be true immediately after the loop, since nothing
           has happened but leaving the loop.
         - By the exit condition, we know that count = p
         - By substitution, we know that answer = n^p
         - So the postconditions are true.

     Thus, if the preconditions are true, the code is executed,
     and the code terminates, then the postconditions will be true.
```

Question 4
[10 marks]

Complete the following OOT function, whose purpose is to compute the total number of characters in a binary tree of strings.

Hints: (1) Use recursion. (2) The built-in OOT function `length(s)` returns the number of characters in the given string s.

```
type treeNode :
    record
        word : string
        left, right : ^treeNode
    end record

function numChars (root : ^treeNode) : int

    if root = nil then
        result 0
    else
        result length (root -> word) +
            numChars (root -> left) + numChars (root -> right)
    end if

end numChars
```