Santa Monica College
CS 20A, Fall 2014

Programming Project 4 - Due: Sunday, November 23 at 11:59 PM

It's a quiet day day at the office and you are discussing binary trees with your boss. You tell her you just learned about trees in the course and that they are so easy and everything can be implemented with just a few lines of code by taking advantage of their recursive structure. She agrees, but cautions you that some of these very simple recursive methods look a lot simpler when they are given to you than they do when you have to implement them yourself.

Before you even have a chance to clarify your opinion on the matter, she opens her "Interview Questions" notebook and hands you some samples. She has a simple task for you as well, which you can do after you have looked at these questions.

Part 1:

You will implement 3 recursive methods that operate on binary trees (any binary trees, not necessarily balanced, not necessarily BSTs). All methods take a pointer to the root node. The Node struct has been given to you. The code is minimal but some thinking is required.

a) Implement a method that finds the number of nodes in the longest path in the given tree.

```
int longestPath(Node* root) { … }
```

As an example, for a full tree with 3 nodes this would return 2, but for a degenerate tree with 3 nodes looking like a linked list it would return 3.

b) Implement a method that sets the data member of all the nodes in the tree to the size of the subtree rooted at that node.

```
void setSize(Node* root) { … }
```

As an example, for a full tree with 3 nodes this would set the data member of the root node to 3 and the data member of each child to 1.

c) Implement a method that changes the tree as if it was flipped left to right.

```
void flip(Node* root) { … }
```

As an example, the transformation below would be performed.

```
        1                              1
    2      3            to          3     2
  4   5                                 5    4
```

Part 2:

You will implement a method that reads in the given students file and counts how many times each **first** name occurs. It then prints in alphabetical order only the first names that occur at least as often as the "threshold" parameter and their occurrence count, one name-count per line. The higher the threshold parameter, the less names will meet it and be printed. Threshold 1 will print all the names.

```
void printFrequentNames(string filename, int threshold) { … }
```

e.g.:

```
Alice 25
Bob 22
Daniel 15
Jane 35
```

Hints:

Part 1: The code is very simple if you understand recursion and the tree recursive methods we saw in class. Make sure you test your methods well. You can build your own trees to pass to the methods by linking together nodes to form trees of different shapes and sizes. For b) and c), the pre-order traversal might be useful to verify that the method performed what you expected.

Part 2: We will use as a sorted symbol table the *map* class from the STL.

http://www.cplusplus.com/reference/map/map/

Do not be intimidated by the large number of functions you see there; you will only need to use the most basic ones that we have seen and be able to iterate through the map. Here is a simple example:

http://www.cplusplus.com/reference/map/map/begin/

We will see symbol tables in a little more detail the Nov 19 lecture so some questions you may have now will likely be clear then.

Deliverables:

You should submit a zip file named project4_first_last.zip (where first and last are your first and last name) containing **ONLY the file below.**

CS20AProject4.cpp

How you get points:

Part 1 – a, b, c	22 point each
Part 2	34 points

How you lose points:

- You change the given methods' signatures and your code does not compile when I run it. **You should only fill in the empty methods. Can use helper methods if you need to, but do not change the given methods' signatures.**
- If your solution is grossly inefficient. Your solution should be **efficient to a level seen in class for similar problems**.
- You submit your whole workspace or executable files. **Submit only the files the project asks for.**
- If any of your code prints anything at all on the console except for the Part 2 output. **Remove all your print outs, debug statements, etc. Clean up your code and do not leave clutter behind.**
- Your code has no comments where needed. **Comment your code appropriately.**