Lab 10 Recursion versus Iteration & Code Reuse

Goals-
Implement linear data structures using link lists

You will be working with recursive and iterative implementation of Fibonacci Number algorithms. You will be measure the running time of each and compare them.

You do not need to implement the algorithms! Feel free to find the source online, or in a book. Just ensure you cite the source! Here is one site that discusses both versions:

http://www.codeproject.com/Tips/109443/Fibonacci-Recursive-and-Non-Recursive-C

I am not endorsing this site; it is simply the first I found that integrates both. It also demonstrates the use of other topics from the course, such as exceptions.

You must modify the program to measure the running time. You can use this reference to measure execution time: http://www.cplusplus.com/reference/ctime/clock/. You can use any other timing mechanism you chose (other than wall clock time). Your TA(s) may suggest another method (many have their own favorite). As discussed in class, the difference should be visible quickly so you won't need very precise measurements.

Before you start write down your expectations.

Then compile and run your program(s). Record your measurements present then with your analysis.

So far you've done double recursion and iteration. At the bottom of this page there is sample code for recursive factorial and tail-recursive factorial functions. Add these functions to your program or write a new program (classes are not required). Insert the timing code and compile and run your program. To take advantage of the tail recursion you'll need to turn some level of optimization on. Use the man pages on flip or do online research to find out what you need. Compare the results of iteration, tail-recursion, single recursion, and double recursion. Are they what you expected?

HINT: Start with small values of N.

NOTE: An important skill in software development is doing research for ideas or known solutions to problems. In this case the simple searches were done for you. But get in the habit of including citations in the comments for each file! The CodeProject site is a good example of licensing. It is clear that others can use this code as long as they do not claim it as their own. An important point when borrowing code!

Modular Grading

We are using modular grading. Each lab will be divided into specific modules. Each module will be graded pass/fail. It either works properly or it does not. 10% of every lab or assignment grade is style/comments or other elements of self-documenting code and clarity. Remember the labs are worth 10 points total.

Programming style- 1 point

Modify the main function to measure the run time of each algorithm- 3 point

Save the measured times for analysis- 3 point

Analysis and discussion of results- 3 points

Code from the slides. Feel Free to use something else.

Not Tail Recursive

```
public long rfactorial (int n) {
     if (n == 1)
          return 1;
     else
     return n * rfactorial(n-1);
}
```

Tail Recursive

```
public long factorialHelper (int n, int result) {
     if (n == 1)
          return result;
     else
     return factorialHelper(n-1,n*result);
}

public long factorial(int n) {
     return factorialHelper(n,1);
}
```