ECE 3020 Homework 2
Due Date: Friday, September 8, 3:00 PM

Consider the following mathematical sequence:

$$F(n) = F(n-1) + F(n-3), \qquad F(0) = 0,\ F(1) = 1,\ F(2) = 2$$

The running time of a straightforward recursive implementation of this sequence grows exponentially, just like the recursive Fibonacci. Using C/C++, Java, or Python, write a recursive function to compute this sequence that works in the following way. The function should be similar to a straightforward recursive implementation except that it will eliminate redundant recursive calls. The first time F(i) is called, F(i-1) and F(i-3) should be recursively called, as in the normal version. However, when F(i-1) and F(i-3) return and are added together, the program should store the F(i) result in an array before returning. Subsequent calls to F(i) should detect that the function has already been invoked at least once for that i and read the value out of the array rather than making recursive calls to F(i-1) and F(i-3).

Time the execution of your recursive function and compare it against a straightforward recursive version and an iterative version (these can be generated with trivial modifications to the recursive and iterative Fibonacci code available on the course Web site). Be sure to write the two comparison functions in the same language you used for your recursive version to make a fair comparison. Calculate the running times for n=10 up to n=50 in increments of 10. Be aware that the straightforward recursive version might take a few minutes for n=50.

For your recursive version, turn in the source code and a sample output showing the final computed value and the sequence of function calls made during execution for n=10. Also turn in a table of execution times for the three programs over the specified range of n values.

Guidelines and hints for measuring CPU time: Make sure to measure the CPU times of the programs, *not* the wall clock times. You can do this with a system call, which will depend on the particular system you are using. When measuring performance, you should delete any output statements, because they can have a significant performance impact. Finally, if the running time is too small, you will not be able to get an accurate measurement and many CPU times might appear to be exactly the same. If you observe this, insert a loop in your program to run the program a large number of times and then divide by the number of iterations to get an individual CPU time value.