CSCI-21 Programming assignment 8 — recursion using a full stack frame — due 5/1/19

You should write these in C++ or some other block structured procedural language first, then translate them into MIPS assembler. Use the "real-world" calling convention from lecture.

Exercise 1:

Write a recursive function using a "real-world" stack frame to implement the Fibonacci function, with all parameters passed on the stack, not via registers. Allocate local variables on the stack, not in the data segment. For this exercise, the Fibonacci series is 0, 1, 1, 2, 3, 5, 8 ... and the elements are numbered from 0, so fib(6) = 8.

Your pseudocode for this function will be like this. The parameter `n` must be held on the stack, not in a register. The variables `x` and `y` must also be local variables allocated on the stack.

```
int Fib( int n )
{
   if( n <= 0 )
      return 0
   endif
   if( n == 1 )
      return 1
   endif
   x = Fib( n-1 )
   y = Fib( n-2 )
   return x + y
}
```

The return value will be in $v0. Test up to Fib(10).

Exercise 2:

Various infinite series for calculating Pi have been discovered. The first such series found was the Gregory-Leibniz series:

$\pi = 4 * (1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots)$

This series is of little practical value because enormous numbers of terms are required to achieve a good approximation. However, it does provide good programming practice.

Write a SPIM program that writes out the sum of the first 1000, 2000, 3000, ..., 10000 terms of the series (e.g., count to 10000 stopping to save the current value after every 1000 passes through the loop or recursive calls). Write both iterative and recursive versions of the function to approximate pi. Have the program call the iterative version then the recursive version, saving the results in arrays, and then print the corresponding values from the two calls, first iterative then recursive, one pair per line, with a tab between the values and a newline after the last value on the line.