CSCI-21 Lab #8, due 4/20/16

Use the "real-world" stack frame-based linkage convention (parameters and local variables on the stack) for this.

Part 1:
Write a program that computes the value of triangle numbers using a recursive subroutine. The definition of a triangle number is:

Triangle(N <= 1) = 1

Triangle(N > 1) = N + Triangle(N-1)

For example:

Triangle(1) = 1

Triangle(2) = 3

Triangle(3) = 6

Triangle(4) = 10

For this, each function invocation has its own parameter N in its stack frame. Then it gets the return value from the next invocation and adds it into N (stored in the stack frame) before retrieving N and putting the result into $v0 to return to the caller. I know you don't actually need to store it first, but that's how a real compiler will likely do it.

Your program will prompt the user for N and calculate and print Triangle(N). Test with several values, including a large value like 100.

Part 2:
The square of an integer N is equal to Triangle(N) + Triangle(N-1). Write a recursive function using the Triangle function from part 1 to calculate Square(N). Write a program to prompt for N and print the result of Square(N). The definition of a square number is:

Square(N <= 1) = 1

Square(N > 1) = Triangle(N) + Triangle(N-1)

Call Square using a full stack frame, and have Square call Triangle twice, passing its argument on the stack. For the recursion, create automatic local variables on the stack and save both recursive results in the variables, then retrieve the values to add together to return to the caller.

For both programs, use addu to ignore the errors caused by potential overflow. You may assume that the user will enter a reasonable value greater than or equal to 1. Test with several values, including a large value like 100.