CSCI-21 Lab #6, due 4/20/22

Use the "frame-based linkage convention" (argument in $a0 register, local variables on the stack) for this. Since the subroutine might need to save the argument, save it into a local variable (on the stack) in the callee prolog, and re-load it in the callee epilog.

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 argument n which is passed in $a0, and will be stored in its stack frame. Then it gets the return value from the next invocation and adds it to its n (as stored in the stack frame) before putting the result into $v0 to return to the caller. Store and re-load $a0 even if you are in the base case.

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, n > 1, 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 stack frame, and have Square call Triangle twice, passing its argument in the $a0 register. For the recursion, create automatic local variables on the stack for the argument and 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 relatively large value like 100.