Assignment: Write a recursive program that calculates Fibonacci function for an integer.

Details: You program must be recursive and must use this algorithm:
   if (n == 0) return 0
   else if (n == 1) return 1
   else return fib(n-1) + fib(n-2).

The main program should prompt for the integer, verify it is non-negative, and then call the function fib(n).

The register usage and the stack usage must confirm to what is given in the partial code below.

Verification: Verify using SPIM that it works for all cases, when the input is negative, zero and positive.

Include comments where appropriate.

Submit using WebCT.

Partial Code:

# Description: Computes the Fibonacci function using recursion.
# Input: n, which must be a nonnegative integer.
# Output:  fib(n).
#
# Main program should do this:
#  print prompt
#  call fib(read) and print result.
# Register usage:
#  $a0 = n (passed to fib)
#  $s1 = f(n)
    .data
    .align 2
# Prompts
This program evaluates the Fibonacci function.
Enter value for n: 
The number is not non-negative.

```
.text
.align 2  #aligns the next instruction on a word boundary
.globl main
main:
# Print the prompts
 li $v0, 4
 la $a0, prmp1
 syscall
 li $v0, 4
 la $a0, prmp2
 syscall
# Read n, verifies it is non-negative and call fib with result
# Your code here

# Print result
 li $v0, 4
 la $a0, desc
 syscall
 li $v0, 1
 move $a0, $s1
 syscall
# Exit
 li $v0, 10
 syscall

# Algorithm for Fib(n):
#   if (n == 0) return 0
#   else if (n == 1) return 1
#   else return fib(n-1) + fib(n-2).
#
# Register usage:
#   $a0 = n (argument)
#   $t1 = fib(n-1)
#   $t2 = fib(n-2)
#   $v0 = 1 (for comparison)
#
# Stack usage:
# 1. push return address, n, before calling fib(n-1)
# 2. pop n
# 3. push n, fib(n-1), before calling fib(n-2)
# 4. pop fib(n-1), n, return address
fib:     bne $a0, $zero, fibn0     # if n == 0 ..
         move $v0, $zero         # .. return 0
         jr $31
fibn0:   # n != 0
         li $v0, 1
         bne $a0, $v0, fibn1     # if n == 1 ..
         jr $31                # .. return 1
fibn1:   # n > 1
## Compute fib(n-1)
   addi $sp, $sp, -8     # push ..
   sw $ra, 4($sp)      # .. return address
   sw $a0, 0($sp)      # .. and n
   addi $a0, $a0, -1   # pass argument n-1 ..
   jal fib           # .. to fib
   move $t1, $v0      # $t1 = fib(n-1)
   lw $a0, 0($sp)   # pop n
   addi $sp, $sp, 4   # ... from stack
## Compute fib(n-2)
## Your code here
# Now return fib(n-1) + fib(n-2)
# Add your code here
    # $v0 = fib(n) = fib(n-1) + fib(n-2)
    # return to caller