```
# Program to calculate the factorial of a number
# Uses function "factorial" that takes integer N as user input where N > 0, displays result (N!)

# 2 methods for error-checking:
#   1 - require N to be >= 1
#   2 - allow N to be anything but return result 0 for N < 1

# Written by Kollen G


        .data
        .align  2
prompt: .asciiz "Enter integer to compute factorial: "
display:.asciiz "The computed factorial is: "
error:  .asciiz "Error: integer must be 1 or above.\n"

#--------------------------------
        .text
        .globl  main

main:
    move    $s0, $0     # s0: computed factorial to display = 0


# -----------------------------------------------------------
# method 2: accept any value for N but return 0 if N < 1
# -------------------------------------
    la  $a0, prompt #load prompt string
    li  $v0, 4      #code to print string
    syscall         #print

    li  $v0, 5      #take int input
    syscall
    move    $s1, $v0    # s1 = user input

    blt     $s1,1,print # if input < 1 then skip factorial calc


#------ Set up function call
continue:
    move    $a0, $s1    # a0 stores the user input

    jal factorial   # v0: computed factorial value
    move    $s0, $v0    # s0: computed factorial to display


#------ Display results and exit ---------------------------------

print:
    la  $a0, display    #load display string
    li  $v0, 4      #code to print string
    syscall         #print

    li  $v0, 1      #code to print int
    move    $a0, $s0    #load computed factorial
    syscall         #print


#------------------ Exit ---------------------
        li  $v0, 10
    syscall
#*******************************************************************


#*******************************************************************
    # factorial function
    #
    # a0 - value of user input
    #
    # v0 - computed n factorial

factorial:
#---------------- Usual stuff at function beginning  ---------------
        addi    $sp, $sp, -24   # allocate stack space for 6 values
        sw  $ra, 20($sp)    # store off the return addr, etc
        sw  $s0, 16($sp)
        sw  $s1, 12($sp)
        sw  $s2, 8($sp)
        sw  $s3, 4($sp)
        sw  $s4, 0($sp)

#-------------------------- function body -------------------------
    move    $s0, $a0    # s0: user input int
    move    $s1, $0
    addi    $s1, $0,1   # s1: product = 1

fLoop:  ble $s0,1,fDone # while (N > 1)
    mul $s1,$s1,$s0 # product = product * N
    addi    $s0,$s0,-1  # N--
    J fLoop

#------ Return computed factorial value
fDone:  move    $v0, $s1

#--------------------- Usual stuff at function end -----------------
        lw      $ra, 20($sp)    # restore the return address and
        lw  $s0, 16($sp)    # other used registers...
        lw  $s1, 12($sp)
        lw  $s2, 8($sp)
        lw $s3, 4($sp)
        lw $s4, 0($sp)
        addi   $sp, $sp, 24
        jr      $ra        # return to the calling function
#*******************************************************************


```