Assignment Instructions:

1) The Factorial

The factorial of a non-negative integer n, denoted by $n!$, is the product of all positive integers less than or equal to n. Enclosed is a simplified version of the MIPS assembly language recursive implementation of the factorial function. Trace the factorial example carefully using QTSPIM

2) Recursive definition of multiplication

The function $rmult(a, b)$ for two positive integers $1 \leq a$, and $1 \leq b$, is defined as the following: $rmult(a, 1) = a;$ $rmult(a, b) = a + rmult(a, b - 1)$

Write a recursive version of $rmult()$ in C or C++ and a pseudo C program (based on chapter 2 in the book) then use these programs to develop a MIPS program that gets as input two integers $0 < a \leq 255$, and $0 < b \leq 255$, and returns the result of $rmult(a, b)$ in $v1.

```
########################################################################
#    Functional Description: Main program to test Factorial function
########################################################################

           .data
           .align          2

           .text
main:      addiu      $sp,  $sp,  -8          # Allocate space
mloop:

           li         $v0,  4                 # Get value for N

           sw         $v0,  0     ($sp)
           jal        Fac                     # Call factorial
           or         $v1,  $v0,  $0          # Result in $v1
           addiu      $sp,  8                 # De-allocate space

           li         $v0,  10
           syscall
########################################################################
# Functional Description: Recursive Factorial Fac  (N:  in,  N! :out)
########################################################################
Fac:
      lw          $a0,  0     ($sp)


      addiu       $sp,        $sp,  -16       #   Allocate
      sw          $ra,        12    ($sp)     #   Save return address
      sw          $a0,        8     ($sp)
      slti        $t0,        $a0,  2         #   If N is 1 or 0,  then
                                              # return the value 1
     beqz         $t0,        Go
     li           $v0,        1
     b            facret
Go:
      addi        $a0,        $a0,  -1
      sw          $a0,        0     ($sp)     #   Pass N-1 to factorial
      jal         Fac                         #   Recursive call
      lw          $v0,        4     ($sp)     #   Get  (N-1)  !  back.
      lw          $ra,        12    ($sp)
      lw          $a0,        8     ($sp)
      mult        $v0,        $a0             #  N* (N-1)  !
      mflo        $v0
facret:
      addiu       $sp,        $sp,  16        #  De-allocate
      sw          $v0,        4     ($sp)
      jr          $ra
```