```
#Sawyer Fenwick 6005011
#COSC 2P12 Assign_3
#PART C
#This program accepts one positive integer and determines the number of possible permutations
#based on the formula P(n,r) = n!/(n-r)!

.data
msg: .asciiz "Please Enter A Positive Integer 'n' For The Equation P(n,r) = n!/(n-r)!: "
msg2:    .asciiz "Error: Number Entered Is Negative."
str:    .asciiz "P("
str2:   .asciiz ","
str3:   .asciiz ")= "
    .text
main:
    la $a0, msg #load a0 with contents of msg
    li $v0, 4   #syscall print_str
    syscall

    li $a0, 10  #load a0 with newline char
    li $v0, 11  #syscall print_char
    syscall

    li $v0, 5   #syscall read_int
    syscall

    move $t1, $v0   #t1 = input (n)

    bltz $t1, error #if input is not positive throw error

    addi $t1, $t1, 1 #add 1 to n so n = true input in inner loop
outer:
    beqz $t1, exit  #if n = 0, no more calculations end program

    subi $t1, $t1, 1    #reduce n by 1
    move $t2, $t1   #t2 = r
inner:
    beqz $t2, outer     #if r = 0 jump back to outer loop

    jal pnr     #calculate pnr
    jal print   #print result

    subi $t2, $t2, 1    #reduce r by 1
    b inner     #return to inner loop
exit:
    li $v0, 10  #syscall exit
    syscall
pnr:
    addi $sp, $sp, -4   #adjust stack pointer to store return address and argument
    sw $ra, 0($sp)      #save ra

    move $a0, $t1
    jal factorial   #jump to factorial
    move $s1, $v0   #s1 = n!

    sub $t3, $t1, $t2   #t3 = n - r

    move $a0, $t3
    jal factorial   #jump to factorial
    move $s2, $v0   #s2 = (n-r)!

    div $s3, $s1, $s2   #s3 = n!/(n-r)! (final answer)

    lw $ra, 0($sp)      #load old ra for return
    addi $sp, $sp, 4    #adjust stack pointer back
    jr $ra
factorial:  #base case of factorial function
    addi $sp, $sp, -8   #adjust stack pointer to store return address and argument
    sw $s0, 4($sp)      #save $s0 onto the stack
    sw $ra, 0($sp)      #save $ra onto the stack
    bne $a0, 0, else
    addi $v0, $zero, 1  #return 1
    j return        #jump to return
else:
    move $s0, $a0       #backup a0
    addi $a0, $a0, -1   #reduce a0 by 1
    jal factorial       #jump to factorial
    mult $s0, $v0       #return n*fact(n-1)
    mflo $v0
return:
    lw $s0, 4($sp)      #load s0 off the stack
    lw $ra, 0($sp)      #get ra off the stack
    addi $sp, $sp, 8    #put the stack back
    jr $ra          #return
print:
    la $a0, str #load a0 with contents of str
    li $v0, 4   #syscall print_str
    syscall

    la $a0, ($t1)   #load a0 with integer n
    li $v0, 1   #syscall print_int
    syscall

    la $a0, str2    #load a0 with contents of str2
    li $v0, 4   #syscall print_str
    syscall

    la $a0, ($t2)   #load a0 with integer r
    li $v0, 1   #syscall print_int
    syscall

    la $a0, str3    #load a0 with contents of str3
    li $v0, 4   #syscall print_str
    syscall

    la $a0, ($s3)   #load a0 with result of pnr
    li $v0, 1   #syscall print_int
    syscall

    li $a0, 10  #load a0 with newline char
    li $v0, 11  #syscall print_char
    syscall

    jr $ra      #return
error:
    la $a0, msg2    #load a0 with contents of msg2
    li $v0, 4   #syscall print_str
    syscall

    li $a0, 10  #load a0 with newline char
    li $v0, 11  #syscall print_char
    syscall

    b main      #return to main, ask for input again
```