```
1:  # Program to calculate the factorial of a number
2:  # Uses function "factorial" that takes integer N as user input where N > 0, displays r
esult (N!)
3:
4:  # 2 methods for error-checking:
5:  #   1 - require N to be >= 1
6:  #   2 - allow N to be anything but return result 0 for N < 1
7:
8:  # Written by Kollen G
9:
10:
11:         .data
12:         .align  2
13: prompt: .asciiz "Enter integer to compute factorial: "
14: display:.asciiz "The computed factorial is: "
15: error:  .asciiz "Error: integer must be 1 or above.\n"
16:
17: #-------------------------------
18:         .text
19:         .globl  main
20:
21: main:
22:     move    $s0, $0     # s0: computed factorial to display = 0
23:
24:
25: # ------------------------------------------------------------
26: # method 1: require user to enter N >= 1 & inform error message
27: # -------------------------------------
28:     addi    $s1,$0,0    # s1: user input = 0
29: loop:   bge     $s1,1,continue  # (while s1 < 1)
30:     la  $a0, prompt #load prompt string
31:     li  $v0, 4
32:     syscall
33:
34:     li  $v0, 5      #take int input
35:     syscall
36:     move    $s1, $v0    # s1 = user input
37:
38:     blt     $s1,1,err   # check if (input < 1) then branch
39:     J loop
40: err:    la  $a0, error  #load & print error string
41:     li  $v0, 4
42:     syscall
43:     J loop
44:
45:
46: #------ Set up function call
47: continue:
48:     move    $a0, $s1    # a0 stores the user input
49:
50:     jal factorial   # v0: computed factorial value
51:     move    $s0, $v0    # s0: computed factorial to display
52:
53:
54: #------ Display results and exit ---------------------------------
```

```
55:
56: print:
57:     la  $a0, display    #load display string
58:     li  $v0, 4      #code to print string
59:     syscall         #print
60:
61:     li  $v0, 1      #code to print int
62:     move    $a0, $s0    #load computed factorial
63:     syscall         #print
64:
65:
66: #----------------- Exit ---------------------
67:         li  $v0, 10
68:     syscall
69: #*****************************************************************
70:
71:
72: #*****************************************************************
73:     # factorial function
74:     #
75:     # a0 - value of user input
76:     #
77:     # v0 - computed n factorial
78:
79: factorial:
80: #--------------- Usual stuff at function beginning  ---------------
81:         addi    $sp, $sp, -24   # allocate stack space for 6 values
82:         sw  $ra, 20($sp)    # store off the return addr, etc
83:         sw  $s0, 16($sp)
84:         sw  $s1, 12($sp)
85:         sw  $s2, 8($sp)
86:         sw  $s3, 4($sp)
87:         sw  $s4, 0($sp)
88:
89: #-------------------------- function body -------------------------
90:     move    $s0, $a0    # s0: user input int
91:     move    $s1, $0
92:     addi    $s1, $0,1   # s1: product = 1
93:
94: fLoop:  ble $s0,1,fDone # while (N > 1)
95:     mul $s1,$s1,$s0 # product = product * N
96:     addi    $s0,$s0,-1  # N--
97:     J fLoop
98:
99: #------ Return computed factorial value
100: fDone: move    $v0, $s1
101:
102: #-------------------- Usual stuff at function end -----------------
103:         lw     $ra, 20($sp)    # restore the return address and
104:         lw $s0, 16($sp)    # other used registers...
105:         lw $s1, 12($sp)
106:         lw $s2, 8($sp)
107:         lw $s3, 4($sp)
108:         lw $s4, 0($sp)
109:         addi   $sp, $sp, 24
```

```
110:         jr      $ra         # return to the calling function
111: #****************************************************************
112:
113:
```