```
#########################################################
#   equal.asm                                           #
#   compares two numbers                                #
#########################################################
.data
number: .asciiz "Give a number: "
difftxt: .asciiz "Numbers are different\n"
sametxt: .asciiz "Numbers are the same\n"

.text
la $a0, number
li $v0, 4
syscall                   # print string
li $v0, 5
syscall                   # read int into $v0
move $t0, $v0
li $v0, 4
syscall                   # print string
li $v0, 5
syscall                   # read int into $v0
move $t1, $v0

li $v0, 4
beq $t1, $t0, same        # if ($t1==$t0)
different:                # false
  la $a0, difftxt
  syscall
  j terminateprog
same:                     # true
  la $a0, sametxt
  syscall
terminateprog:
li $v0, 10
syscall                   # terminate program
```

```
#########################################################
#   greater.asm                                          #
#   compares two numbers                                #
#########################################################

.data
number: .asciiz "Give a number: "
difftxt: .asciiz "Smaller or equal\n"
sametxt: .asciiz "Greater\n"

.text
la $a0, number
li $v0, 4
syscall                  # print string
li $v0, 5
syscall                  # read int into $v0
move $t0, $v0
li $v0, 4
syscall                  # print string
li $v0, 5
syscall                  # read int into $v0
move $t1, $v0

li $v0, 4
bgt $t1, $t0, greater      # if ($t1>$t0)
smallerorequal:              # false
  la $a0, difftxt
  syscall
  j terminateprog
greater:                   # true
  la $a0, sametxt
  syscall
terminateprog:
li $v0, 10
syscall                  # terminate program
```

```
#########################################################
#   dowhile.asm                                         #
#   implements a do-while loop                          #
#########################################################

.data

inputtxt: .asciiz "Give a number: "
singlespace: .asciiz " "

.text

repeat:
#    read integer into $t0:
la $a0, inputtxt
li $v0, 4 # print string
syscall
li $v0, 5 # read integer into $v0
syscall
bne $v0, $zero, repeat

endprog:
li $v0, 10
syscall
```

```
#####################################################################
#  for.asm                                                          #
#  MIPS assembler program that shows how to implement a for loop    #
#    for (i=0; i<n; i++)  printf("%s", benfica);                    #
#####################################################################

.data
prompt: .asciiz "Give a number: "
benfica: .asciiz "Benfica, o glorioso!\n"

.text
la $a0, prompt
li $v0, 4
syscall                   # print string
li $v0, 5
syscall                   # read int into $v0
move $t0, $v0

# we will implement the C code:
#     for (i=0; i<$t0; i++)
#         printf("%s", benfica);

  la $a0, benfica
  li $v0, 4
  move $t1, $zero           # initial value of i=0
startloop:
  beq $t1, $t0, exitloop  # jump if end value already reached
  syscall                 # printf
  addi $t1, $t1, 1        # i++
  j startloop             # go back to start of loop
exitloop:
li $v0, 10
syscall                   # terminate program
```

```
##########################################
#  factorialfor.asm                      #
#  MIPS assembler program that shows an  #
#    example of a for-loop for n!        #
##########################################

.text

main:
  li $t0, 5  # calculating 5!
  li $a0, 1  # storing the result
loop:
  beqz $t0, exitloop
  mul $a0, $a0, $t0
  addi $t0, $t0, -1
  j loop
exitloop:   #printresult (in $a0):
  li $v0, 1
  syscall

# terminate program:
  li $v0, 10
  syscall
```