```
############################################################################
# Noah del Angel, CS 2318-002, Assignment 2 Part 1 Program C
############################################################################
# Allocate a global array (i.e., space in the data segment) enough for storing 4
# integers and initialize the array (from 1st to 4th element) with 8, 1, 3 and 2
# at the same time (i.e., DON'T first allocate uninitialized space for array and
# later write code to put the values into array).
#
# Display a labeled output about the array's initial contents (from 1st to 4th element).
# IMPORTANT (for the purpose of this exercise):
#               You are to load the values of the array elements from memory and use those
#               values to generate the labeled output. (You are not to simply display a
#               hard-coded, preset string.)
#
# Re-order the values in the array so that the contents of the array in memory
# (from 1st to 4th element) becomes 2, 3, 1 and 8, using the following operations
# in the order listed (to not defeat the goals of this exercise, you must NOT
# change the specified operations and order, even if  doing so will accomplish the
# specified operations and order, even if  doing so will accomplish the same
# effect more efficiently):
#               Swap the contents in memory of the 1st and 4th elements of the array.
#               Swap the contents in memory of the 2nd and 3rd elements of the array.
#
# IMPORTANT (for the purpose of this exercise):
# When performing each of the two swap operations above, you can re-use (where
# expedient) the array's base address in register (loaded when performing the
# display of the array's initial contents) but you MUST re-load the values of the
# values of the associated array elements fresh from memory (i.e., assuming no
# knowledge that certain values might have already existed in some registers due
# to prior operations).
#
# Display a labeled output about the array's contents (from 4th to 1st element)
# after the 2 swapping operations above.
#               NOTE: It is from 4th to 1st element and not from 1st to 4th element.
#
# IMPORTANT (for the purpose of this exercise):
# When displaying the after-swap labeled output, you can re-use the array's base
# address in register (loaded when performing prior operations) but you MUST
# re-load the values of the array elements fresh from memory (i.e., assuming no
# knowledge that certain values might have already existed in some registers due
# to prior operations).
#
# CAUTION:
# Too many past students regretted having points taken off for not labeling
# output.
############################ data segment ################################
               .data
intArr:        .word 8, 1, 3, 2
initialCons:   .asciiz "Initial Conditions for the array: "
resultCons:    .asciiz "Conditions after changing the array: "
############################ code segment ################################
               .text
```

```
              .globl main
main:
              #Load array into $t0
              la $t0, intArr

              #Load data to print
              lw $t1, 0($t0)
              lw $t2, 4($t0)
              lw $t3, 8($t0)
              lw $t4, 12($t0)

              #print array
              li $v0, 4
              la $a0, initialCons
              syscall

              li $v0, 1
              move $a0, $t1
              syscall

              move $a0, $t2
              syscall

              move $a0, $t3
              syscall

              move $a0, $t4
              syscall

              #print a new line
              li $v0, 11
              li $a0, '\n'
              syscall

              #swap 8 and 2
              lw $t1, 0($t0)
              lw $t2, 12($t0)


              sw $t1, 12($t0)
              sw $t2, 0($t0)



              #Reload array into $t0
              la $t0, intArr

              #swap 3 and 1
              lw $t1, 4($t0)
              lw $t2, 8($t0)

              sw $t1, 8($t0)
              sw $t2, 4($t0)
```

```
#Reload array into $t0
la $t0, intArr

#Load data to print
lw $t1, 0($t0)
lw $t2, 4($t0)
lw $t3, 8($t0)
lw $t4, 12($t0)

#print array
li $v0, 4
la $a0, resultCons
syscall

li $v0, 1
move $a0, $t4
syscall

move $a0, $t3
syscall

move $a0, $t2
syscall

move $a0, $t1
syscall
```