```
# MIPS Implementation of selection sort
# Brandon Ingli
# 4 March 2019

.data
prompt1:  .asciiz "Enter number of elements: "
prompt2:  .asciiz "Enter elements one per line:\n"
newline:  .asciiz "\n"

      .align 2 #Make sure our words line up appropriately

list: .space 400 #Space for up to 100 words

#Begin Code
.text

# n -> $s0
# Base(list) -> $s1
# i -> $s3
# j -> $s4
# min_pos -> $s5
# temp -> $s6

main:
      la $s1, list #Load address of list into $s1

      # Prompt for number of elements
      la $a0, prompt1 #load address of prompt 1 into first argument
      li $v0, 4 #load print string syscall code
      syscall #make the syscall

      #Read number of elements, store in $s0
      li $v0, 5 #read int syscall code
      syscall #read int
      move $s0, $v0 #move read int into $s0

      # Prompt for elements
      la $a0, prompt2 #load address of prompt 1 into first argument
      li $v0, 4 #load print string syscall code
      syscall #make the syscall

      #Loop 1: Read ints into array
      #i = 0
      li $s3, 0
for1:
      bge $s3, $s0, for1_exit #"for i < n;" branch if i >= n
```

```

        #Calculate address of list[i]
        sll $t0, $s3, 2 #t0 = i * 4 for offset
        addu $t0, $t0, $s1 #to is now &list[i]

        #read integer
        li $v0, 5 #read int syscall code
        syscall

        sw $v0, 0($t0) #store that integer into list[i]

        addi $s3, $s3, 1 #i++
        j for1 #loop back

for1_exit:
        #Loop 2: Outer loop of sort
        li $s3, 0 #i=0
for2:
        addi $t0, $s0, -1 #$t0 = n-1
        bge $s3, $t0, for2_exit #"for i < n - 1;" branch if i >= n-1

        move $s5, $s3 #min_pos = i

        # Loop 3: Inner loop of sort
        addi $s4, $s3, 1 #j = i + 1
for3:
        bge $s4, $s0, for3_exit #"for j < n;" branch if j >= n

        #Load list[j]
        sll $t0, $s4, 2 #t0 = j * 4 for offset
        addu $t0, $t0, $s1 #t0 = &list[j]
        lw $t0, 0($t0) #t0 = list[j]

        #load list[min_pos]
        sll $t1, $s5, 2 #t1 = min_pos * 4 for offset
        addu $t1, $t1, $s1 #t1 = &list[min_pos]
        lw $t1, 0($t1) #t1 = list[min_pos]

        #if list[j] < list[min_pos]
        bge $t0, $t1, if1_exit #"if list[j] < list[min_pos];" branch when list
        move $s5, $s4 #min_pos = j
        # No need to jump since there's no else
if1_exit:
        addi $s4, $s4, 1 #j++
        j for3 #loop again on inner loop
for3_exit:
        #swap
        #temp = list[i]
        sll $t0, $s3, 2 #t0 = i *= 4 for offset
        addu $t0, $t0, $s1 #t0 = &list[i]
        lw $s6, 0($t0) #temp = list[i]

        #list[i] = list[min_pos]
        sll $t1, $s5, 2 #t1 = min_pos * 4 for offset
        addu $t1, $t1, $s1 #t1 = &list[min_pos]
        lw $t2, 0($t1) #t2 = list[min_pos]
        sw $t2, 0($t0) #list[i] = list[min_pos]

        sw $s6, 0($t1) #list[min_pos] = temp


        addi $s3, $s3, 1 #i++

        j for2 #loop again on outer loop
for2_exit:
        #print "\n"
        la $a0, newline
        li $v0, 4
        syscall

        #Loop 4: Print out contents
        li $s3, 0 #i = 0
for4:
        bge $s3, $s0, for4_exit #"for i < n;" branch when i >= n

        #Calculate &list[i]
        sll $t0, $s3, 2 #t0 = i * 4 for offset
        addu $t0, $t0, $s1 #t0 = &list[i]

        #Print list[i]
        lw $a0, 0($t0) #load list[i] into first argument
        li $v0, 1 #set print int syscall code
        syscall

        #print "\n"
        la $a0, newline
        li $v0, 4
        syscall

        addi $s3, $s3, 1 #i++

        j for4 #loop again

for4_exit:
        jr $ra #exit
```