6. In this question you will be asked to write a small subroutine using MIPS assembler. You will then write a second program that calls this subroutine more than once. *A copy of Appendix-B of your text book containing all MIPS Instructions has been provided to you.*

   (a) (9 points) Write a subroutine called `findmin` that will return the **minimum** value of an array. The location of the array in memory (`a0`) and the length of the array (`a1`) will be passed as parameters. The minimum value will be returned in the register `v0`.

Solution:

```
1  findmin:    lw $t4, 0($a0)          # t4 is minimum
2              addi $t1, $0, 0         # loop counter t1 init 0
3
4  loop:       addi $t1,$t1,1          # t1 ++
5              beq $t1, $a1, done      # loop reaches a1 --> done
6              sll $t2,$t1,2           # byte addressing, multiply
7              add $t2,$t2,$a0         # address of $t1 th member
8              lw $t3,0($t2)           # load value from memory
9              slt $t5, $t4,$t3        # compare to $t4
10             beq $t5,$0,updatemin    # t3 is smaller
11             j loop                  # repeat
12
13 updatemin: add $t4,$0,$t3           # update $t4
14             j loop                  # continue loop
15
16 done:       add $v0,$0,$t4          # move result to $t4
17             jr $ra                  # jump to $ra
```

(b) (6 points) Now that you have the subroutine `findmin`, write a small MIPS assembly subroutine that:

- finds the minimum of a first array of 64 values starting from the address `0x0000 0400`
- finds the minimum of a second array of 64 values starting from the address `0x0000 0824`
- jumps to label (`first`) if the minimum value of the first array is greater than the minimum value of the second array otherwise execution jumps to label (`second`)
- At the end, jump back to the calling program
- If necessary, save values in stack before calling `findmin`.

Solution:

```
1  sol:   addi $sp, $sp, -4       # make room on stack
2         sw   $ra, 0($sp)        # save ra
3
4         addi $a0, $0, 0x0400    # first address
5         addi $a1, $0, 64        # number of elements
6         jal  findmin            # v0=findmin(a0,a1)
7         add  $s1,$0,$v0         # save result to $s1
8
9         addi $a0, $0, 0x0824    # second address
10        addi $a1, $0, 64        # number of elements
11        jal  findmin            # v0=findmin(a0,a1)
12
13        slt  $t0, $s1, $v0      # is $s1 less than v0
14        beq  $t0, $0, first     # no : jump to first
15
16 second:                        # do something
17        j    end                # jump over first
18
19 first:                         # do something
20
21 end:   lw   $ra, 0($sp)        # restore ra
22        addi $sp, $sp,4         # restore stack
23        jr   $ra                # jump to $ra
```