

Name:

Grade:

/20

[20] 2) The program of this exercise deals with arrays of numbers and subroutines. Next page is the program outlines. At the beginning of your program you will allocate empty storage for two original arrays and their sorted versions. For the overall program layout, use the program skeleton file (lab2-v??-A-skl) available on Canvas.

[16] 2.a) Complete this MSP430 assembly language program where the **SORT1** section sets the R4/R5/R6 parameters, which are used by the **COPY** and **SORT** subroutines to copy and sort array **ARY1**. R4 holds the starting address of the array. R5 holds the length of the array. R6 holds the starting location of the sorted array. **COPY** subroutine copies the contents of array **ARY1** into **ARY1S**. **SORT** subroutine sorts the elements on **ARY1S** in place. **SORT2** section is similar to **SORT1** above using same registers.

Arrays are in decimal notation! Sort Arrays in ascending order from lowest to highest value.

Main Program: [6] for Program setup, and [10] for Sort Subroutine.

~~Do not change~~ Use the following values for the ~~ary~~ elements. If the values in the skeleton code are different, use these values.

ARY1: (10, 33, -91, -75, 82, 11, -28, -99, 31, -92, 80),

ARY2: (10, 21, 22, 20, -49, -80, 32, 62, 60, 61, -82)

[4] 2.b) Run your program and verify the results by using the Memory Browser window in the CCS Debug view. Write the Hex Values in order:

ARY1S: 0A, 90, A4, A5, B5, E4, 0B, 1F, 21, 50, S2

ARY2S: 0A, AE, B0, CF, 14, 18, 16, 20, 3C, 3D, 3E

Name:

Grade:

/20

```

;----- Your Sorting lab starts here -----
;Memory allocation of Arrays must be done before the RESET and Stop WDT
ARY1      .set    0x0200      ;Memory allocation    ARY1
ARY1S     .set    0x0210      ;Memory allocation    ARYS
ARY2      .set    0x0220      ;Memory allocation    ARY2
ARY2S     .set    0x0230      ;Memory allocation    AR2S

        clr    R4          ;clearing all register being use is a good
        clr    R5          ;programming practice
        clr    R6

        mov.w #ARY1, R4    ;initialize R4 as a pointer to array1
        mov.w #ARY1S, R6   ;initialize R4 as a pointer to array1 sorted
        call  #ArraySetup1;then call subroutine ArraySetup1
        call  #COPY         ;Copy elements from ARY1 to ARY1S space
        call  #SORT         ;Sort elements in ARY1

        mov.w #ARY2, R4    ;initialize R4 as a pointer to array2
        mov.w #ARY2S, R6   ;initialize R4 as a pointer to array2 sorted
        call  #ArraySetup2;then call subroutine ArraySetup2
        call  #COPY         ;Copy elements from ARY2 to ARY2S space
        call  #SORT         ;Sort elements in ARY2

        Mainloop   jmp   Mainloop    ;Infinite Loop

        ArraySetup1 mov.b #10, 0(R4)    ;Array element initialization Subroutine
        mov.b #__, 1(R4)    ;First start with the number of elements
        mov.b #__, 2(R4)    ;and then fill in the 10 elements.
        .....
        ret

        ArraySetup2 .....
        ret

        COPY      .....
        ret          ;Copy original Array to allocated Array-
                    ;Sorted space

        SORT      .....
        ret          ;Subroutine SORT sorts array from
                    ;lowest to highest value

;----- Your Sorting lab ends here -----
;
```