# **PEX 3: Basic Assembly Code**

### *Due Lesson T29 (1630 04 Apr 12)*

**Help Policy**

**AUTHORIZED RESOURCES:** Any, EXCEPT another cadet’s program.

**NOTE:**

* Never copy another person’s work and submit it as your own.
* You must document all help received from sources other than your instructor.
* DFCS will recommend a course grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others.

1. Requirements

This PEX is designed to ensure that you have a working knowledge of how to write and debug a program that uses recursive procedures and manipulates data in memory. You will write an assembly language program that sorts data in memory.

Your program should:

* Accept a list of up to number 20 number
* Sort the numbers in-place using the provided recursive algorithm
* Print out the results

Your program should be written using the MARS simulator and should be named **PEX3\_Your Last Name.s**. Use functional decomposition to create at least 4 separate procedures (with arguments):

* InputData (Takes address of where to store an integer and Prompts the user to add a number, returns 1 if the number was non-zero and 0 if the user typed in “0”).
* SortArray (takes the memory location of the start of an array of integers, the index of the left most portion of the array to sort, and the index of the right-most portion of the array to sort. It then sorts the array)
* PrintIntArray (takes the staring address of an Integer Array and the number of element to print and prints it to the screen)

You are not limited to these three procedures. You can add more as you see fit.

1. Turn In Requirements

Deliverables

1. An Assembly Code File.
2. Documentation Statement.
3. Sample Run

The following is an example run of the program with user inputs in bold:

Enter a number**:** **29**

Enter a number**:** **63**

Enter a number**:** **5**

Enter a number**:** **238**

Enter a number**:** **4**

Enter a number**:** **19**

Enter a number**:** **2**

Enter a number**:** **94**

Enter a number**:** **0**

The sorted list is: 4, 5, 19, 29, 63, 94, 238

1. Pseudo Code Of Sort Array

The following is the pseudo code that you should use as the basis for your SortArray implementation

**function** SortArray(Array, left, right)

**if** left < right

**integer** middle *=* floor((left + right)/2)

**integer** leftIdx *=* left

**integer** rightIdx= right

**integer** pivotValue= Array[middle]

**while** leftIdx **<** rightIdx

**while** array[leftIdx] **<** pivotValue **and** leftIdx **<** rightIdx

leftIdx = leftIdx + 1

**while** array[rightIdx] **>** pivotValue **and** rightIdx **>** leftIdx

rightIdx = rightIdx - 1

**if** leftIdx < rightIdx

**swap** array[leftIdx] **with** array[rightIdx]

**if** rightIdx < leftIdx

**swap** leftIdx **with** rightIdx

SortArray(Array, left, leftIdx)

SortArray(Array, (leftIdx + 1), right)

**return** **PEX 3 Cut Sheet (75 Points)**

|  |
| --- |
| **Name:**  **Grade: /75** |

|  |  |  |
| --- | --- | --- |
| **Requirements** |  |  |
| **Implementation** | Main function correctly calls all the other methods | 15 |
|  | Reads in data using InputData procedure | 15 |
|  | SortArray functions correctly | 15 |
|  | SortArray uses the stack so that it can be called recursively | 15 |
|  | Displays the contents of the array using the PrintIntArray method | 15 |
| **Design** | Poorly Commented | -(5-20) |
|  | Poor Organization | -10 |
|  |  | **(75)** |
| **Penalties** |  |  |
|  | **Vague/Missing Documentation (5%):** | **−** |
|  | **Late Penalty Cap ( %):** | **−** |
|  |  | **−** |

|  |  |  |
| --- | --- | --- |
| **Total** |  | **(75)** |