Programming Project 3

OPEN THIS ASSESSMENT WITH THE LINK ABOVE AND SEE THE ATTACHED GRADING RUBRIC (click the View Rubric button): **Your program begins with 100 points. Each error deducts the indicated points until the score is 0 (if it gets that far). I will reference the Error Number when providing feedback. Once you complete your program, go through this list CAREFULLY and do a self-grading of your program BEFORE submitting it.**

Programming Project 3 (100 Points toward Course Grade)

Instructions: The following programming problem can be solved by a program that uses three basic tasks- Input Data, Process Data, and Output Results. To process the data, it uses loops, arrays, decisions, accumulating, counting, searching and sorting techniques. Use **Python** to write a suitable program to solve this problem.

Problem Statement

Assume the Scores array is parallel to the Players array (both arrays are below).

Scores array

Scores[0] = 198

Scores[1] = 486

Scores[2] = 651

Scores[3] = 185

Scores[4] = 216

Scores[5] = 912

Scores[6] = 173

Scores[7] = 319

Scores[8] = 846

Scores[9] = 989

Players Array

Players[0] = "Joe"

Players[1] = "Ann"

Players[2] = "Marty"

Players[3] = "Tim"

Players[4] = "Rosy"

Players[5] = "Jane"

Players[6] = "Bob"

Players[7] = "Lily"

```
Players[8] = "Granny"

Players[9] = "Liz"
```

Write a looping program that presents the user with 3 options:

1) Sort Output by Players

2) Sort Output by Scores

3) Exit Program

When the first option is selected, sort the Players array in alphabetical order, keeping the Scores array parallel. Add code that determines the highest and lowest scores in the list. Include code to display each player's score and name in the sorted order. Below the sorted list display the highest and lowest scores in the list and the name of the player who received that score.

Your sort by Player output display should look like this:

```
Scores Sorted by Player:

486     Ann

173     Bob

846     Granny

912     Jane

198     Joe

319     Lily

989     Liz

651     Marty

216     Rosy

185     Tim

-----------------------------------

989     Highest Score by Liz

173     Lowest Score by Bob
```

When the second option is selected, sort the Scores array in numerical order, keeping the Players array parallel. Add code that determines the average score of the entire list. Include code to display each player's score and name in the sorted order. Below the sorted list display the average of all scores in the list. Your sort by Scores output display should look like this:

```
Players Sorted by Scores:
```

```
173    Bob

185    Tim

198    Joe

216    Rosy

319    Lily

486    Ann

651    Marty

846    Granny

912    Jane

989    Liz

---------------------------

498    Average Score
```

You may use either the Bubble Sort or the Selection Sort algorithms.

Option three is self explanatory. NEVER call "main" from inside your program (other than at the very end of your Python program) to keep the programming running. Use a loop that keeps your program running until the user chooses option 3.
Round the Average score to the nearest whole number, as shown in the output example above.
You MUST use **Modular Programming** techniques by using Sub Modules in your program. Your "main" module should not be very large. Again, NEVER call "main" from inside your program. Also, do NOT use "recursion" in this program (submodules that call themselves). You are only allowed to use looping techniques to repeat sections of your submodules.
You may NOT "hard code" the numbers for highest and lowest scores. These must be discovered through algorithms.
You may NOT "hard code" the number for the average score. Accumulate the scores in a loop then calculate the average. If the array data is changed, the Hi/Low/Avg scores should automatically be found or calculated with the new data.
Hard-code the values of the arrays into your program. Do NOT ask the user to input the values.

Other Requirements:

- Documentation: Use "Comments" to document your code.
- Test and debug your Program: Create sample input data, run the program, then check your answers with a calculator or Excel. If something did not match up, then fix your program.
- Program must execute and produce correct output.
- Read this page again to be sure you covered all requirements.
- See the Programming Project Rubric for grading principles.
- Extra Credit: 1) Add an option to the menu with code that allows the user to type in a Player's name and then displays the Player's score. 2) Use files to input your array data.

Submission Instructions:

- You must submit a **Python program** file. Your Python file will be the *.py* file created when you save your project.
- Name the Python file (replacing LastName and FirstInitial with YOUR name): **LastName_FirstInitial_Program_3.py** (example: **Smith_J_Program_3.py**).
- Attach your file to your assignment submission upload. If you find that you made an error and want to resubmit before the due date, you may do so. However, only the LAST file uploaded by the due date will be graded.