PROGRAMMING PROJECT#4: INFORMATION AND SUBMISSION FORM - The "Score Analysis" program (50pts) - due NO LATER than Monday May 4, 2015*

This assignment information sheet is to be used ONLY by Santa Rosa Junior College students currently enrolled in CS 10 Sections 4293, 4524 and 5393 (Spring 2015).

Prior to working on this programming project....please read Gaddis textbook Chapter#8 to understand the set-up, structure and typical types of processing done with **User- Defined Single or Multi-Dimensional arrays** and review Chapter 9.1 - 9.4 for additional content on **Searching and Sorting arrays**. You may skip your reading of section 8.11 - "Introduction to STL vector and section 9.5 - "Sorting and Searching STL Vectors" during your review of the content provided in these two chapters. STL vectors are a more advanced topic and beyond the scope of the current CS10 Title V course outline.

It is recommended that you also read the weekly lesson notes on **Single Dimensional Arrays** on the class schedule pages, review any chapter resources provided on the **class calendar** for weeks#15 - 16. Check the **class downloads and resources page** for any additional ancillary material or project related "hints" provided by your instructor on the current topic under discussion for the week.

Originality of Work : Projects and assignments that show evidence of lack of originality of work in any of the following forms will receive a grade of 0.
Collaboration: Although discussing assignments and projects with fellow students is encouraged, these discussions should not be about specific code and one student should never look at another student's actual code. If significant portions of your code are similar to another student's code, both parties will receive a grade of 0 on the assignment. This is not an hypothetical situation. It happens every semester. If you find yourself hopelessly stuck, get help from me. In summary, there are three things that you should never do:

1. look at another student's code,
2. let another student look at your code, or
3. discuss code with another student in enough detail that parts of your program look identical.

Previous Semester Coursework: You should also never look at a solution to a programming exercise or a project from a previous semester, whether it is a sample solution or a student submission.
Other Sources: Many of the programs that you do this semester are classic problems that have been solved by thousands of students, and some may be found in various textbooks. If you find a book (or website, or other source) that includes a solution or partial solution to one of our assignment, you may review the source code but do not simply copy/paste the code into your program. Instead recreate your own algorithm and source code prior to submitting the assignment or project.

Instructions to all sections: Use your compiler to write a single C++ program following the program specifications detailed below. After successfully compiling and debugging (if needed) your program, paste your source code and contents of your program output in the text input areas provided for the programming segments of this assignment. Complete all parts of this programming assignment and check for errors, prior to clicking submit.

Please make sure you have correctly input your name, section#, email address, SID (last four digits of your srjc student identification number) and the correct web form code visible to you. Upon submission you should receive a confirmation message. If you need further clarification please DO NOT hesitate to contact me, click to **email me** .
IMPORTANT NOTE*: Please make sure Project#4 is submitted no LATER than May 4, 2015. NO exceptions!

STUDENT NAME: **CS10 SECTION#:**

EMAIL ADDRESS: **SID#:** (last four digits)

Project#4- 1-D array processing: Score Analysis program

Write a program that gives the user a menu of 8 options. The options are:

1. Read in 10 scores (integers) from the user.
2. Read in 10 scores from a file, scores.txt.
3. Print the 10 scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the mean (average) of the 10 scores.
7. Print a score based on an entry or row# and show how many scores are higher.
8. Exit the program.

- Both the size of the array (a constant) and the array itself must be declared locally inside of int main(). You should initialize the array to some value (probably 0) in case the user chooses options 3 - 7 before 1 or 2. Options 1 or 2 will overwrite any value previously stored in the array.
- Each option in the menu must simply call a function. Each function must at minimum pass in the array and the size of the array. You must list all function prototypes above int main () and all function definitions must appear after int main(). You will find a few helpful programs in Gaddis text - Chapter 8 that provide examples on how to declare and process homogenous data with 1-D arrays as well as the set-up of some useful array functions.
- You may use the input file, provided here => **scores.txt** or you may create your own. You do not need to turn in your input file. Please note that I will be using this version of the scores.txt when testing your program.
- Provide appropriate input validation where necessary, use meaningful identifier names and DO NOT declare any variables globally.

Here is a sample run of the program that you may use as a guide in helping set-up the interface and also provides a few check figures...

```
 ------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:1

Enter score #1: 56
Enter score #2: 89
Enter score #3: 78
Enter score #4: 66
Enter score #5: 72
Enter score #6: 99
Enter score #7: 95
Enter score #8: 100
Enter score #9: 85
Enter score #10: 68

------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
```

```
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:4

The highest score is 100

--------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
--------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:5

The lowest score is 56

--------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
--------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:6

The average score is 80.80

--------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
--------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:2

Please enter filename:scores.txt
File has successfully opened and 10 scores have been read and stored.
Please select the print all scores menu option to view the scores
```

```
--------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
--------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:3

score #1:       47
score #2:       89
score #3:       65
score #4:       55
score #5:       62
score #6:       95
score #7:       87
score #8:       75
score #9:       100
score #10:      79

--------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
--------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:7

Please enter entry or row # of score you want:7
Entry #7 score:87
Score statistics:3 higher

--------------------------------------------------
1-D ARRAY PROCESSING MENU OPTIONS
--------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:9
INVALID CHOICE ...please retype
--------------------------------------------------
```

```
1-D ARRAY PROCESSING MENU OPTIONS
-------------------------------------------------
1. Read in 10 scores from user.
2. Read in 10 scores from the file, scores.txt.
3. Print all scores.
4. Print the highest score.
5. Print the lowest score.
6. Print the average score.
7. Print one score (give its entry number)
8. Quit program

Enter your choice:8

Array processing test now concluded. Exiting program .....

Process returned 0 (0x0)   execution time : 128.997 s
Press any key to continue.
```

If your program does not compile, it will not be graded. HINT: I am thinking seven to eight array processing functions with procedures or algorithms that address the menu choice should be adequate. Use a do while loop with a switch statement. You should call the relevant functions from each case. Compile your code often. Only write a small portion of code, let's say a single function for a specific menu option, check that it still compiles before moving on. This way when you get a syntax error, you can be fairly certain the error is in the part you just wrote.

EXTRA CREDIT (15points) - Applies only if programming segment (i.e., menu options 1-8) above is successfully completed.

Modify the program that you wrote above "Score Analysis" so it includes two more menu options and write the necessary procedures specified below in two additional functions:

9. Sort scores in descending order (use a selection sort algorithm - see Gaddis textbook Section 9.3 pgs 605 - 613 or review the selection sort example in section 10.6 of the Lesson notes for week 7-8) - 8 points

10. Search for a score (use a binary search algorithm - see Gaddis textbook Section 9.1 pgs 595 - 602) - 7 points

---

Paste your program "source code" for Project#4 - Score Analysis program in the text input area below:

Paste your program output for Project#4 - Score Analysis program in the text input area below:

General comment area: You may use the text area below to provide any comments on this programming project or if you are unclear and have a question on some of the new C++ syntax or semantics related content covered in recent weeks.

Please also include any comments on something new and useful (C++ related) that you discovered while developing and implementing algorithms (source code) to complete this array processing project. *THANK YOU! (Comments are for Instructor use only and are not used for grading purposes)*

Enter web form code Z6NS2

reload image

RE-TYPE EMAIL ADDRESS:

Submit Reset

Sujan Sarkar - CS Instructor, Santa Rosa Junior College

Updated Sunday April 19, 2015