Programming Project 3 Rubric

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.

Error Number	Error Description	Points Deducted
A. Program Execution and Calculated Results		
0	Program did not Execute (run). If the program does not run, I will *not* debug it to find out why.	**-100**
1	Program Executes (runs), but produces incorrect output. Be sure to check your results with a calculator before finalizing the program. Also, look at your values in red in the Watch window. This will tell you if your variables contain the correct values needed for correct final results. I will *attempt* to tell you why the output was incorrect.	**-50**
2	Program Executes (runs), but produces incorrect output. Did not swap the parallel array when swapping the one being sorted on. When sorting more than one array, the programmer must swap all arrays using the same index in the first swap. See the textbook section "8.5 Focus on Problem Solving: A grade management Program." Among other concepts that it breaks down and explains, page 498 shows the logic on sorting 4 parallel arrays. Notice lines 10 through 30. When a necessary swap is found in the Finals array in line 11, all 4 arrays are swapped using the exact same index. This keeps all 4 arrays' data matched up by the index that is used for the record as seen in the table on page 492. When you, as the programmer, move one item of data in one array, you must also move each of the other 3 items on that row to the new location (index) to keep the record intact	**-40**
3	Program Executes (runs), but produces partial incorrect output. Sorted properly, but High/Low/Average output was incorrect.	**-30**
4	Program Executes (runs), but was confusing to use. This could be due to extra (not required) input requests that are never used or asking for user input of the final results that were to be calculated by the program.	**-20**
5	No menu presented to user.	**-20**
6	Created your own menu. Did not use the example in instructions.	**-10**
7	Program did not loop back after asking user if they wanted to continue or exit the program. See example attached to Lesson 4's reading activity.	**-10**
8	Though a looping program was attempted, program would not exit properly. See example attached to Lesson 4's reading activity.	**-10**
9	Were to Hard-Code the 10 players and scores found in the instructions. You asked the user to input all this data.	**-10**
10	Did not properly check and handle menu selection input errors that were "out of bounds". You should have used loops immediately after collecting input from the user to check for valid input and to keep prompting the user	**-5**

	for correct input. This is demonstrated several times within chapter 5. Page 310 shows this technique with the guessing example.	
11	You hard-coded your average calculation (divided by 10) instead of counting players within the loop and dividing by the "count." What if more players were added? You would need to recompile your entire program after editing it.	**-5**
12	Hard-coded Highest score by using index 10 after sorting. What if more players were added to the program?	**-5**
13	One or both of your sorted displays were in descending order and was supposed to be in ascending order. See the example output on the instructions.	**-5**
14	Hard-coded Average score. Did not calculate this.	**-5**
15	Hard-coded Highest score. Did not calculate this.	**-5**
16	Hard-coded Lowest score. Did not calculate this.	**-5**
17	High/Low output was to be on the Sort by Players output. You placed it on the Sort by Scores report.	**-4**
18	Average output was to be on the Sort by Scores output. You placed it on the Sort by Players report.	**-4**
19	Average should have been rounded up to the nearest whole number as shown in the example output seen in the instructions.	**-2**
B. Output in MasterConsole Window		
0	Though the calculated output was correct, you did not tell the user anything about the output displayed. Learn to label and format your output for user understanding. There are plenty of examples in the textbook for proper output display.	**-20**
1	No welcome message telling the user what the program did. See page 126 of the textbook.	**-10**
2	Welcome message did not provide enough information to user.	**-5**
3	Did not place any space between the labels and numbers in your output statements. Make sure your output is formatted correctly in the future. Example: *Correct output* "Amount 200" | *Incorrect output* "Amount200"	**-4**
4	Did not place any space between the Scores and Names.	**-4**
5	Welcome message had more than 1 grammar or spelling error.	**-4**
6	Incomplete output. See requirements.	**-4**
7	You should not take the "End Current Line" option off of so many of your output symbol boxes. This created output to the MasterConsole that was difficult to follow.	**-2**
8	Your Welcome message should have been in the MasterConsole, not in a prompt. See the examples in the textbook.	**-2**
9	Should have sent menu to MasterConsole window, then used a shorter prompt for choice.	**-2**
10	Did not repeat display of menu for user to see after each run of selected routine.	**-2**
C. Internal Documentation (Comments)		
0	No comments used anywhere in program. See requirements.	**-20**
1	Not enough comments used in the program. See requirements.	**-10**

2	Did not tell what "type" each variable was in your comments, per instructions.	**-10**
3	Did not tell what "type" each variable was in your comments, per instructions. Simply stating it is "number" does not tell the "type."	**-10**
4	Most of your comments were not meaningful to the program. Your comments should relate to the program you are creating and NOT to the assignment instructions or my requirements. Your comments for this course should be specific to what the program code is doing. For example: "1 of 3 inputs required" does not tell another programmer what this input is all about. For this course, your comments should be as if you are creating these programs for an actual job.	**-10**
5	Labeled some of your variables with incorrect types. Review all of the pseudocode examples and chapter 2 in the textbook again for correct names for typing your variables and know when to correctly use each type.	**-5**
6	More than 2 misspellings in comments.	**-5**
7	Learn to press ENTER on your comments to keep them from running off the screen to the right. Break the one long line up into several shorter lines.	**-2**
8	No need to identify the type of flowchart symbol being used in your comments. The shape of the symbol is self-documenting, which is one reason we use flowcharts.	**-1**
9	No need to place // before comments in flowchart. They are already identified as a comment by the symbol used.	**-1**
10	Left empty comment symbols in program. Delete these before submitting.	**-1**
D. Input Prompts		
0	Prompts were inadequate. Need more than just variable names or blank prompts with just an empty entry box. See examples in the textbook and read about what makes a good prompt. You must communicate with your user better.	**-10**
1	Sent all Prompts to the MasterConsole window with nothing in the Prompt boxes themselves.	**-5**
2	Prompts to user had more than 1 grammar or spelling error.	**-4**
3	Some prompts were too long. All of it did not display in the dialog box and could not be read by the user. If you need to display more detailed information in your prompts, give the more detailed information in the MasterConsole window with a shorter prompt in the input box. Be sure to test your program for things like this in the future.	**-2**
4	You do not need to duplicate your Prompts in both the input box AND the MasterConsole window. Use the MasterConsole window for your "output" to the user. The prompts are already seen in the dialog boxes where the user enters the inputs.	**-1**
E. Modular Programming		
0	No modular programming used.	**-30**
1	Program either called the Main module or used some form of recursion (modules that call themselves or repeatedly call each other without returning control to the calling module).	**-30**
2	Good try at modular programming design. You need to study the textbook examples more closely and show me that you are learning what true	**-10**

	modular programming is at this point in the course. This only tells me that you are not using the textbook examples to model and hone your programming skills. You should have had at least 4 modules other than main as demonstrated in the textbook models: Welcome, Input, Calculate, and Output.	
3	Some or all module names were not meaningful. Their names should reflect what the modules actually do.	**-4**
4	You made this program much more complex than it needed to be. You repeated a lot of code rather than keeping the design simple. Study the examples in the textbook and practice creating them before attempting the Final Programming Project.	**-2**
F. Proper Submission of Assignment		
0	Submitted RAPTOR backup file. I had to delete the extra file extension before I was able to open the program. Be sure to submit correct file in the future. You may need to change the setting on your computer to allow you to see file extensions.	**-10**
1	Did not name submitted file according to instructions.	**-4**
2	Variable names were not meaningful. See page 35 of textbook.	**-4**
G. Extra Credit		
0	Used methods taught in future lessons of course in this program and used them CORRECTLY.	**+10**
1	Added search routine correctly.	**+5**
2	Used files to input the values of your arrays correctly.	**+5**