Zak Attack

Professor Caleb Fowler

Sp16

Problem.

Professor Zak allows students to drop the three lowest scores on the nine! 100-point quizzes she gives during the semester. Design and code an application that accepts a student name and the nine quiz scores. Output the student's name and the total points for the six highest scoring quizzes. Then output the corresponding letter grade and words of advice.

Constraints.

- Use Structured Programming Techniques.
- Allow name to accept first plus last name (ie 2 words).
- Range test the input data for the grades.
- Create a table to match the numeric score with a letter grade (find an appropriate scale on the Internet).
- Use a parallel array to map the letter grade with an appropriate (unique) word(s) of advice.
- I will compile it with the following
  `g++ -std=c++11 -g -Wall <filename.cpp>`

Bonus Features.

- Add a novel programmer defined feature of your choosing (counts as 2 bonuses!).
- If you want to add other data structures, first, keep the structures specified in the assignment. Then, add the data structure you want at the bottom and print the results again using the new structure. I'm not going to give you any suggestions so you get 2 points for whichever structure(s) you choose to use.

Turning In Your Assignment

Bring working code to class, as you will be peer reviewing it. Once it is successfully reviewed, attach the Peer Review section to your source file. Answer the questions and submit the code to dropbox on D2L. This means that the only code I will have on dropbox is code that you have verified as working. You can also use this as a means of keeping track of what you have turned in - if it's not in dropbox, you didn't turn it in. Note: You can ONLY peer review in class!

UPLOADING, DO NOT save your code as a .cpp file! Save it as a .txt file instead. Don't zip or otherwise compress your files. I will be able to read them once you get them on D2L. I have a script which converts the files to .cpp and automatically executes them.

TURN IT IN by uploading to the D2L Dropbox folder for the appropriate assignment. You do not need to put your name in the **filename**; Homework1, 2 whatever will be just fine. D2L appends student information to the files when I download them, so I will see all this information automatically.

Using the Work of Others.

This is an individual assignment, you may use the Internet and your text to research it, but I expect you to work alone. Copying code from someone else and turning it in as your own is plagiarism. However, you **may** discuss code and the assignment. I have opened discussion groups in D2L to do this. I will monitor this, but not interfere. D2L will check your code against a database of other assignments. It tells me how similar your code is to someone else's. I consider isomorphic homework to be plagiarism. Do your own work.

Discussion.

The purpose of using a standardized file format is so the class can share questions. This will give you another method to prepare for the weekly quizzes. If you do decide to share questions, you will need some way of double checking the answers to make sure they are correct - studying incorrect answers is a waste.

Rubric for Evaluating this Assignment.

Grading Rubric					
	Sophisticated	Highly Competent	Competent	Not Yet Competent	Unacceptable
Solution Fit with Client Needs	As Highly Competent, but also successfully performs 3 bonus features (for a total of 4).	As Competent but also successfully performs 1 bonus feature also	Successfully accomplishes all specifications and constraints with the test data set.	Accomplishes some specifications and/or constraints with test data set. May have logic errors.	Does not meet any specifications or constraints. May not compile.
User Friendliness	~ Code has program greeting to introduce itself. ~ Program identified input expected from user.	~ Code has program greeting to introduce itself. ~ Program identified input expected from user.	~ Code has program greeting to introduce itself. ~ Program identified input expected from user.	Program requires omniscient users to divine expected input(s).	Input prompts are just a blinking cursor.
Comments and Documentation	~ Proper program header. ~ Function's properly commented. ~ Comments identify blocks of logically different code, and/or, modifications to formula's are noted. ~ Good use of whitespace.	~ Proper program header. ~ Function's properly commented.	~ Proper program header.	1 Line comment header and/or comments don't match code.	Missing program header, and/or, missing or incoherent comments.