

Assignment 1

Whirlwind C

50 points

Due: Tuesday, February 5, 2008 by 11:59:59 PM PST

Grade Calculator

Write a program that reads in assignment scores, quiz scores, exam scores, and final exam score from the user, then produces a weighted average. The weighted average will be based on what percent each of the four previous categories (assignments, quizzes, exams, final exam) count. When the percentages of each category are added, they total 100%. Assume the following about each category:

- Assignments
 - There are a total of 7 assignments
 - Each assignment is worth 50 points
 - Assignments account for 40% of the course grade
- Quizzes
 - There are a total of 3 quizzes
 - Each quiz is worth 25 points
 - Quizzes account for 10% of the course grade
- Exams
 - There are a total of 2 exams
 - Each exam is worth 100 points
 - Exams account for 25% of the course grade
- Final Exam
 - The final exam is worth 200 points
 - The final exam accounts for 25% of the course grade

First you must obtain the users name, then you must obtain the necessary scores from the user. Once this is done, report the percentage earned in each category followed by the weighted percentage. In addition, report the Grade Point for the course based on the following table:

- 95% and above 4.0
- for each percentage from 94% down to 65%, drop the grade point by 0.1 -- here are some examples
 - 94% - 3.9
 - 90% - 3.5
 - 82% - 2.7
 - 65% - 1.0
- 64% and 63% are a 0.9
- 62% and 61% are a 0.8

- 60% is a 0.7
- below 60% is a 0.0

Sample Run

Welcome to the Grade Calculator Program.
This program will produce a weighted percentage based on the scores earned in four categories: assignments, quizzes, exams, and a final exam. In addition to the weighted percentage, the associated Grade Point will also be reported.

```
Enter score for Assignment 1: 50
Enter score for Assignment 2: 50
Enter score for Assignment 3: 50
Enter score for Assignment 4: 50
Enter score for Assignment 5: 50
Enter score for Assignment 6: 50
Enter score for Assignment 7: 50
```

```
Enter score for Quiz 1: 25
Enter score for Quiz 2: 25
Enter score for Quiz 3: 25
```

```
Enter score for Exam 1: 100
Enter score for Exam 2: 100
```

```
Enter score for Final Exam: 200
```

```
-----
```

Percentage per category:

```
  Assignments: 100%
   Quizzes:     100%
    Exams:      100%
   Final Exam: 100%
```

```
Your weighted percentage is: %100
Your Grade Point is: 4.0
```

Specifics

- Name your C file **gradeCalculator.c** (contains main)
 - Employ modular design practices -- use individual functions for individual tasks – You must have at least 3 additional functions other than main
 - Use a 3 file format. What you call the others is up to you.
 - Try to keep your method small -- if a method is 20 lines of code or more in length it can *probably* be broken into at least two smaller functions
 - Use constants where possible. If you have a number used in your program that can be given a specific name, and it's value never changes, represent it via a constant. This makes your code more self-documenting
 - Negative scores are not possible in the grading scheme so do not allow the user to enter them
 - All scores should be treated as real numbers (doubles), since half points are possible in scores.
 - You may utilize arrays on this assignment.
 - EXTRA CREDIT (5 points possible): Allow the user to specify how many scores there are for each category
 - EXTRA CREDIT (5 points possible): Allow the user to specify the points possible for each assignment
-

To Turn In

Submit a tarball to Blackboard containing **gradeCalculator.c**, your other .c and .h file, and an output capture from running your program that includes data such that the user gets a 4.0, a 2.7, and a 0.7. Name your output capture **gradeCalc_output.txt**. Include comments at the top of your source file that has your name, a description of the program, a history of work done on the program, a list of shortcomings (if any), and a list of extra credit items implemented (if any). Also include brief comments in front of each method you write that describes what the method does (NOTE: if the name you used completely describes what the function does, you do not need to provide any comments). Finally, document any code you feel another programmer viewing your code for the first time might need to help understand something you did.