

Simple Algebraic Calculations

One of the first things you will learn about C++ is how to perform numerical computations. In this project, you are given an incomplete program (see the end of this specification), which already includes the statements necessary to read input and specify output. You must add statements so that your completed program will perform the specified calculations.

The input file:

The input file for this program is named "GradeData.txt". A sample input file is given below. The input file begins with four header lines, which are meaningful to a human reader but must be ignored by the program. The fifth line of the file contains academic performance data for one student for their previous terms. Following the fifth line are two other header lines, which must be skipped by the program. All of the remaining lines of data pertain to a particular course that the student has taken for the current term, which has just ended. Each of these lines provides the course index number, the credit hours for the course and the student's achieved numerical grade.

```
; CS 1044 Fall 2002
; Project 2 Test File 0
; Previous Semesters' Data
; SSN      Qual Cred  Hrs Att
123456789  40.2    15.0
; Current Semester Data
; Index    Hours    Grade
11366     3.0     2.7
12778     3.0     2.3
15566     3.0     3.3
13366     3.0     2.0
11223     3.0     3.0
```

There may be data for an arbitrary number of courses; i.e., we cannot make any assumptions about the minimum or maximum number of courses the student might have taken.

As your class covers input and output in C++, you should examine the given program and make sure you understand precisely how it manages its input and output.

What must be calculated:

From the previous term data, the program must calculate the student's GPA, (Grade Point Average). This will give the student's GPA prior to the beginning of the current term.

The program must calculate quality credits earned for each course in the current semester and summary data for the current term courses. Specifically, the program must compute the total quality points, total hours attempted, and GPA for the current term courses. Additionally, the program must also calculate updated overall academic data for the student. This involves calculating new values for the overall quality credits, hours attempted and a new GPA, which includes the current semester grades.

In order to produce the most accurate results possible in C++, all the decimal values are stored using variables of type `double`, not type `float`.

Although the given program is incomplete, it does declare all the variables that are needed to calculate the required results. You may declare additional variables if you like, but they will not be logically necessary.

The output file:

The output file is named "GPAdat.txt". An output file, which corresponds to the given input file, is shown below. The first two lines specify the programmer and the assignment, and the third is blank. The fourth line specifies column headers for the previous term data. The fifth line echoes the previous term data values and the computed previous GPA.

The sixth line is blank. The seventh line contains table headings for the current term data and the eighth line separates the table data from the labels.

The table body contains one line of output for each course, displaying the index, credit hours, numerical grade and computed quality credits earned for that course. After the last table line, another line of dashes is written for separation. Following the separation line, aligned under corresponding columns are the semester summary data. This is followed by a blank line. The last two lines of output contain labels and the overall academic updated data.

When a decimal number is written, the number of digits displayed after the decimal point is called the precision of the displayed value. The quality credits, hours attempted, credit hours and grades must be written with precision 1, and the GPA values with precision 4 (as shown in the output sample).

```

Programmer: David McPherson
CS 1044 Project 2 Fall 2002

    SSN      GPA    Qual Cred  Hrs Att
123456789  2.6800   40.2      15.0

Index  Hours  Grade  Qual Cred
-----
11366  3.0    2.7    8.1
12778  3.0    2.3    6.9
15566  3.0    3.3    9.9
13366  3.0    2.0    6.0
11223  3.0    3.0    9.0
-----
Sem Hours  Sem Credit  Sem GPA
    15.0      39.9      2.6600

GPA    Qual Cred  Hrs Att
2.6700  80.1      30.0

```

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. In any case, the given code will exactly match the labels and spacing shown above.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://spasm.cs.vt.edu:8080/curator/>

Evaluation:

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. We will not be evaluating your submission of this program for documentation style. However, you should examine the given program as a guide to acceptable documentation, and include similar comments for the statements you add to it.

For a number of the later projects, we will also evaluate your submission for documentation, and for other requirements. It is best to begin preparing for that now.

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
// anyone other than my instructor or the teaching assistants  
// assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
// or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
// was obtained from another source, such as a text book or course  
// notes, that has been clearly noted with a proper citation in  
// the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
// interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.

The Program:

```

// CS 1044 Project 2   Fall 2002
//
// Programmer:      David McPherson
// Last modified:   August 23, 2002
// Compiler:        MS Visual C++ .NET
// Platform:        Pentium 4 1.7 / Windows 2000 5.00.2195
//
// Purpose:
//   This program performs simple computations relating to the computation
//   of A student's GPA. Given the previous academic quality credits & hours
//   attempted and the current term courses, grades and credit hours, the program
//   calculates the student's prior GPA, the quality credits for each semester
//   course, the semester hours attempted, term GPA and quality credits.
//
//   In addition, the program calculates the updated overall academic information:
//   GPA, hours attempted and quality credits.
//
#include <fstream>      // for file streams for input/output
#include <iomanip>      // for formatting manipulators
using namespace std;

int main() {

    ifstream In("GradeData.txt");    // Open the input file.

    ofstream Out("GPAdata.txt");      // Open the output file.
                                     // Enable floating-point output formatting.
    Out.setf(ios::floatfield, ios::fixed);
    Out.setf(ios::showpoint);

    double QualCred,                // overall quality credits
           HrsAtt,                   // overall hours attempted
           gpa;                      // overall Grade Point Average (GPA)

    double Hours,                   // course credit hours
           Grade,                    // course grade
           Credits;                  // course quality credits

    double SemHours,                // total semester credit hours
           SemGPA,                   // Semester GPA
           SemCredits;               // total semester quality credits

    long   ssn,                      // social security number
           Index;                    // course index number

    // Initialize the running semester totals:
    SemHours   = 0.0;
    SemCredits = 0.0;

    // Write the header information to the output file:
    Out << "Programmer: <put your name here>" << endl;
    Out << "CS 1044 Project 2 Fall 2002" << endl;
    Out << endl;
    Out << "   SSN           GPA   Qual Cred  Hrs Att" << endl;

    In.ignore(255, '\n');            // ignore the four header lines
    In.ignore(255, '\n');
    In.ignore(255, '\n');

```

```
In.ignore(255, '\n');

// Read prior academic information :
In >> ssn >> QualCred >> HrsAtt;

In.ignore(255, '\n'); // ignore any remaining data on the line

// Calculate previous GPA:
if (HrsAtt > 0)
    // Some calculations may go here...
else
    gpa = 0.0;

// Write the prior academic data to the output file:
Out << setw( 9) << ssn
    << setw( 8) << setprecision(4) << gpa
    << setw( 8) << setprecision(1) << QualCred
    << setw(10) << setprecision(1) << HrsAtt
    << endl;

Out << endl; // Output a blank line for separation

// Write the table header information to the output file:
Out << "Index Hours Grade Qual Cred" << endl;
Out << "-----" << endl;

In.ignore(255, '\n'); // ignore the two semester header lines
In.ignore(255, '\n');

// Try to read a line of data for a semester course:
In >> Index >> Hours >> Grade;

while (In) {

    // Calculate quality credits for this course:
    // Some calculations may go here...

    // Update the total semester credit hours to include this course:
    // Some calculations may go here...
    // Update the total semester quality credits to include this course:
    // Some calculations may go here...

    // Write the specified course data to the output file:
    Out << setw( 5) << Index
        << setw( 6) << setprecision(1) << Hours
        << setw( 7) << setprecision(1) << Grade
        << setw( 9) << setprecision(1) << Credits
        << endl;

    // Try to read a line of data for another semester course:
    In >> Index >> Hours >> Grade;
}

// Calculate the Semester GPA:
if (SemHours > 0)
    // Some calculations may go here...
else
    SemGPA = 0.0;

// Mark the end of the output table:
Out << "-----" << endl;
Out << " Sem Hours Sem Credit Sem GPA" << endl;
```

```
// Write the semester summary information to the output file:
    Out << setw( 8) << setprecision(1) << SemHours
        << setw(11) << setprecision(1) << SemCredits
        << setw(13) << setprecision(4) << SemGPA
        << endl;

Out << endl; // Output a blank line for separation

// Update the overall quality credits to include this semester:
// Some calculations may go here...
// Update the overall hours attempted to include this semester:
// Some calculations may go here...

// Calculate the overall GPA:
if (HrsAtt > 0)
    // Some calculations may go here...
else
    gpa = 0.0;

// Write the labels and overall summary information to the output file:
Out << " GPA   Qual Cred  Hrs Att" << endl;
Out << setw( 4) << setprecision(4) << gpa
    << setw( 8) << setprecision(1) << QualCred
    << setw(10) << setprecision(1) << HrsAtt
    << endl;

// Close the input and output files:
In.close();
Out.close();

return 0;
}
```