

**CS 1160-01 Introduction to Computer Science
and Programming Methods
Programming Assignment 4
Due, Wednesday, February 16, 2005
A Small Sample Gradebook**

This exercise should provide a little more practice in functional decomposition and will provide a chance to experience file-based input and output. Well, output, anyway.

The problem is to make a small grade report for a fictional class. This class had three in-class tests and a final exam. All these were scored on a 100-point basis. The computation of the final grade weights each *in-class test* at 20% and the *final exam* at 40%. Grading for this course is on a CR/NC basis, i.e., CRedit, No-Credit (also known as “pass/fail”). In order to earn a CR, a student’s weighted average for the term must be greater than 70. (i.e., average > 70 means CR; average \leq 70 means NC).

Your program will accept the information about the individual students from the keyboard and will compile a class report in a *file*. (You may also want the information going into the file to be echoed to your screen, too, but that is optional.)

The input

It should first *prompt the user* for a name to use for the output file.

The program should then accept the information for *many*, i.e., an *unlimited number* of students at any session. It should gather the following information for each student:

last name, first name, netid, test1 score, test2 score, test3 score, final exam score

The first three of these should be **strings** and the last four should be **ints**. Your program should then compute the *final average* as a **float** and determine the student’s term grade. The program should repeat this process so long as the user continues to enter student information. Use a *sentinel* value, such as “done” or “Done” as *last name* to signal the end of the input.

The output

After having received the file name to use for output, your program should open a file output stream (**ofstream**) on the file. It should put out a line of *header labels*, i.e.,

Last Name First Name ID Test1 Test2 Test3 Exam Term Average Grade

After that, whenever the information for a student has been entered and that student’s average and grade computed, your program should write all that information to the file on a single line.

NOTE: You may want to experiment with the formatting capabilities from **iomanip** to set field widths, etc. in order to get nicely readable output.

When the program has completed its work, it should close the file. The output file should now be printable using your favorite program for printing textfiles.

Miscellaneous

Your program should use *functional decomposition* in its design in order to practice with this technique. For example, you could use one function to *gather the input* (Note: *reference parameters* may be useful here.), another function to *compute the term average* (and grade, too?), and another to *write out the desired line* to the output file.

What to turn in

As usual, you should turn in your *source code*, which should be in good style and well commented. You should also turn a record of a good, interactive test session, together with a printing of the resulting file created by your program. This should demonstrate the correct working of your program.