

Functions

Designing and implementing your own functions is the key to producing well-organized, flexible programs in C++. In this project, you will write a short program that will read Curator grading data from an input file, select a specific program grade to assign to a student, and then print nicely formatted results to an output file.

This program will produce a grade report for a single programming assignment. Each student is allowed five (5) submissions. For each student in the input file, select the program submission with the highest grade. In the case of ties, the first submission with the highest grade is used. A “-1” indicates an unused submission. A student entry with a first program grade of -99 signifies the end of the input that should be processed. No output should be produced for this “sentinel” line, or for any subsequent data. The input file format is described below.

The input file:

The input file for this program is named “ProgGrades.txt”. A sample input file is given below:

Boole, George	98	105	-1	-1	-1
Pascal, Blaise	63	48	92	92	92
Babbage, Charles	100	97	100	98	-1
Kepler, Johannes	75	102	100	-1	-1
Clown, Bozo	0	6	6	57	62
Fini, End	-99	-99	-99	-99	-99

Each line in the input file represents the grades for a student. The student’s name appears first, and then a grade for each program submission appears. There will be a student name and five integers on each line. For simplicity, we guarantee that every student will have made at least one submission. We also guarantee that the input file will contain a sentinel line.

This is a tab-delimited file; that is, there is one tab character separating the student name from the first score, and one tab character separating adjacent scores. No input line will contain more than 255 characters.

What must be calculated:

For each student, your program must determine the highest grade and which submission received the highest grade. In the case of a tie, the earliest submission with the highest grade should be selected.

The output file:

The output file is named “GradeReport.txt”. An output file, which corresponds to the given input file, is shown below:

Programmer: Mir Farooq Ali		
CS 1044 Project 5 Summer II 2004		
Student	Submission	Grade
=====		
Boole, George	2	105
Pascal, Blaise	3	92
Babbage, Charles	1	100
Kepler, Johannes	2	102
Clown, Bozo	5	62

The first two lines specify the programmer and the assignment, and the third is blank. The fourth line identifies each column in the grade report. Each of the remaining lines contains the student name, submission number with the highest grade, and the highest grade. The columns in the grade report are separated by a single tab.

Note: When you view the input file or the output file in an editor like *Word* or *Notepad*, the columns may not line up exactly as shown. That is OK. Try opening the file in *Microsoft Excel*, to see how your output is converted to a spreadsheet.

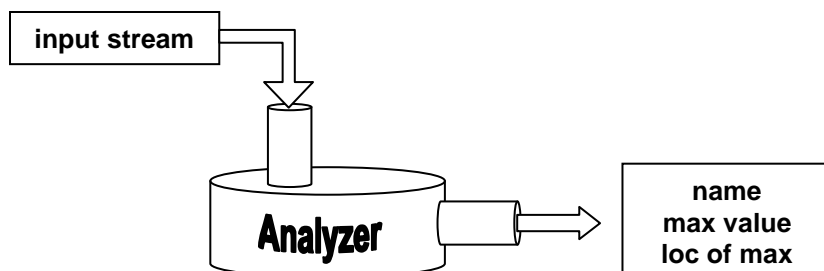
If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You must, however, use blank lines exactly as shown in the sample output.

Function requirements for this program:

Your design must include at least three user-defined functions, not counting `main()`. Since this is the first project in which you have been required to design and implement functions, we will provide some guidance and suggestions.

First of all, you should reconsider the given problem from the beginning. The fundamental idea is to identify what logical tasks need to be carried out, and which of those should be delegated to separate functions. Here are some suggestions...

As was the case with the earlier projects, there are essentially three phases of operation: getting input, performing calculations, and writing output. The input file is read line-by-line; the act of reading a single line of input, and determining the maximum score and its location, could be performed by a function. This "analysis" function would have to be provided with the input stream, and would have to produce the name of the student and the location and value of his/her maximum score. Pictorially, we have something like this:



The "analysis" function must compare values to determine the maximum score. That suggests that one could write a "helper" function that is given two integer values and returns an indication of whether the first is larger than the second.

The writing of output could also be performed by a function (or two). This function would have to be provided with the output stream, and the variables whose values are to be written out.

Documentation and other requirements:

You must meet the following requirements (in addition to designing and implementing a program that merely produces correct output):

- Write a header comment with your identification information, the required pledge statement (below), and a brief description of what the program does.
- Write a comment explaining the purpose of every variable and named constant you use.
- Write comments describing what most of the statements in your program do.
- Use descriptive identifiers for variables and for constants.
- Use named constants instead of "magic numbers" whenever it is appropriate. Note: there are some candidates in this category.
- Use both a `while` loop and a `for` loop (appropriately) in your implementation.
- Design and implement functions, as specified above.
- The first function implemented in your source file must be `main()` – so you must provide appropriate prototypes for each of your functions.
- Each function implementation must be accompanied by a header comment that describes what the function does, the logical purpose of each parameter (including whether it is input, output, or input/output), the pre-conditions that a function call assumes and the post-conditions that are guaranteed, the return value (if any), a list of the functions that call this function, and a list of other functions called by this function (if any). An acceptable sample function header is shown below:

```
// Function name:  Foo()
// Foo() computes the number of items that can be purchased with a given amount
// of money, and adjusts the amount of money available to reflect the purchase
// of that many items.
//
// Input parameters:  (not changed by the function)
//   PricePerItem    price that must be paid for one item
//
// Output parameters: (changed by the function)
//   MoneyAvailable  amount of money that can be spent to purchase items
//
// Return value:
//   The number of items that can be purchased with the given amount of money,
//   at the given price.
//
// Preconditions:
//   PricePerItem is a positive decimal value and MoneyAvailable is a non-
//   negative decimal value.
//
// Postconditions:
//   If PricePerItem is larger than MoneyAvailable, then MoneyAvailable is
//   unchanged and the value zero is returned.
//   Otherwise, the largest value of k such that MoneyAvailable >= k*PricePerItem
//   is determined and returned, and MoneyAvailable is decreased by
//   *PricePerItem.
//
// Called by:  main()
// Calls:      none
//
int Foo(double& MoneyAvailable, double PricePerItem) {

    // function body goes here

}
```

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 submission link can be found at:

<http://eags01.cs.vt.edu:8080/CuratorS04/index.jsp>

Evaluation:

Your submitted program will be assigned a score based upon the runtime testing performed by the Curator System. We may very well evaluate your submission of this program for documentation style, and to see whether you followed the requirements given in this specification. Therefore, you should compare your comments to those given in the programs for projects 1 and 2, as well as consider the scoring of your fourth project. The programs serve as useful a guide to acceptable documentation style at this point in the course.

If your program is evaluated for documentation and requirements, your instructor will specify how that score will be counted.

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.