

## Functions

Designing and implementing your own functions is the key to producing well-organized, flexible programs in C++. In this project, you will re-design your solution to the last project to incorporate functions.

There are NO changes to the functionality specified for Project 6. The input and output file formats remain the same.

**Note:** we are well aware that a student could simply re-submit his/her solution to Project 6 and receive a perfect score from the Curator. However, the TAs will be checking your submission to verify that you have met the requirements of this assignment. The entire point of this project is for you to design and implement your own functions. If the TAs find that you have not done so, we will adjust your score for this assignment to a zero.

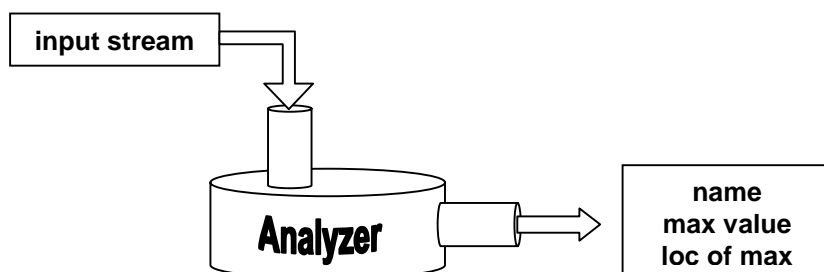
### Function requirements for this program:

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. Your earlier implementation (probably) did not incorporate any user-defined functions, and it is not really a suitable basis for beginning work on this project. That doesn't mean you cannot re-use any of your previous code. To the contrary, the logical design analysis you did for Project 6, and at least some of the implementation you produced, can be recycled into this project. But, you should now consider the design from a different perspective.

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

}
```

**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.

The TAs will also evaluate your submission of this program to see whether you followed the implementation requirements given in this specification. If you do not implement and use functions in your program, your score will be adjusted to zero. The TAs will also check to be sure that you provide header comments for each of your functions, as specified above.

While the TAs will not be conducting any careful examination of your submission for other documentation, you are still expected to follow the same practice specified for all the projects.

Your instructor will specify how the TAs scoring of your submission 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.**