

User-defined Functions:**Detailed Billing for Pa Bell**

For this project you will re-implement the previous project using functions. The input file, most of the required functionality, and the output file will be precisely the same as for the previous project. For that reason, submissions that do not make use of user-defined functions will be assigned a final score of zero.

Input and Output:

See the posted specification for Project 4 for descriptions of these items. There are no changes to those requirements. Sample input and output data will be posted on the course website.

Changes to Calculations:

Most of the calculations performed in Project 4 are unchanged. The only changes are in the rules for calculating the charges for peak and off-peak calls. For off-peak calls, the discount is now applied only if the off-peak minutes exceed 120 (instead of 100). The per-minute rates for off-peak calls are unchanged.

For peak time calls, Pa Bell has instituted an additional charge (a surcharge) if the customer uses too many peak minutes. If the customer accumulates no more than 100 minutes of peak time, the charge is the same as before. If the customer accumulates more than 100 minutes of peak time, there is an additional charge of 5 cents for each minute beyond 100.

Requirement for handling monetary amounts:

In Projects 3 and 4 you were advised to use integers to store monetary values, in order to avoid computational problems that occur when using doubles. For this project, you are explicitly required to implement your solution to use integers to store monetary amounts.

The prices are decimal values and therefore pose some problems that we will discuss in class later. For now, you are required to store monetary amounts as integers, not as doubles. To achieve precisely correct answers, you should read the dollar amount and the cents amount separately and then store the total price in cents. All internal calculations involving money should operate on cents and produce results that are stored as integers. When the time comes to print a monetary amount, you should compute the correct dollar amount and cents amount, as integers, and print those separated by a decimal point. If you do not do this, some of your answers may differ from the correct results in the last digit.

Evaluation:

Everything that was said in the specification for Project 1 about testing still applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

As before, your implementation must meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.

- Use C++ `string` variables to hold character data, not C-style character pointers or arrays.

Reflecting the primary facet of this project, you must make good use of user-defined functions in your implementation. In particular:

- You must implement at least one function that performs input and at least two functions that perform output.
- You must implement at least two functions that compute and return an integer result.
- None of your functions can include more than 30 executable statements. An executable statement is one that causes something to happen. Comments, constant and variable declarations (even if they initialize) do not count as executable statements.
- You must implement a total of at least 5 functions besides `main()`. For reference, my solution contains 13 functions in addition to `main()`.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Every function implementation must have a header comment. The format of the header comment is described in the course notes, as well as on the *Programming Standards* page on the course website.
- Adopt a consistent indentation style and stick to it.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed 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://eags.cs.vt.edu:8080/curator/>

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the pledge statement that was provided with the earlier project specifications in the header comment for your program.

Failure to include the pledge statement will result in a final project score of zero.