Input Failure Control and Decisions: Detailed Billing for Pa Bell

For this project you will modify and extend your implementation for Project 3 to employ an input-failure loop and more sophisticated decisions. The input file will be radically different, but the output file will have precisely the same format.

Sample input data:

Here is a sample input file, named `CallDetails.txt`, for the program. As before, the first three lines specify the name of the customer, the customer's phone number, which is also used as the account number, and the billing date. Each of these lines begins with a label that is terminated by a colon character, followed by an unspecified number of spaces, followed by the relevant data value. The next line will be blank, and the next two will specify column headings and separate them from the body of a table of call data:

```
Name:           John McBryde
Phone:          540.231.2346
Date:           02/24/2001

Type     Number called     Duration
-----------------------------------
P        703.786.4503            33
O        804.331.5604            37
P        703.554.0980             7
O        804.331.5604            32
P        703.786.4503            49
O        804.331.5604            19
O        505.887.9191            71
C        804.331.5604            17
O        703.786.4503            62
C        202.843.0016            48
O        804.331.5604            89
P        703.786.4503            41
C        703.554.0980            27
```

The main part of the input file consists of lines of data, each of which specifies a particular call that was made during the billing period. For each call you will be given a code indicating the type of the call, the number called, and the length of the call in minutes. There are three valid call codes: `'P'` indicates the call was made during peak time, `'O'` indicates the call was made during off-peak time, and `'C'` indicates the call was made using a calling card. The call code is separated from the phone number by a single tab character; the phone number is separated from the length by a single tab character followed by zero or more spaces. There will be a newline character immediately after the length.

The input values are guaranteed to be syntactically and logically correct. The name, phone number and date are just string data. For formatting purposes, you may assume that the name will be no more than 40 characters long, and the phone numbers and date will be no longer than the ones shown above.

Calculations:

As in the previous project, under the calling plan selected by all the customers this project will deal with, Pa Bell has three categories for long-distance calls:

Category	Charge per minute
Peak time	0.10
Off-peak time	0.05
Calling card	0.25

The rates are given in dollars.

Pa Bell has recently added an off-peak time discount for high-volume callers. The first 100 off-peak minutes will be billed at the regular rate given above. However, additional off-peak minutes will be billed at a rate that is $0.02 less. There is no discount for peak minutes or calling card calls.

In addition to the per-minute charges, Pa Bell charges customers on this plan a flat monthly fee of $4.95 unless they make no peak time or off-peak time calls, in which case the flat fee is reduced to $1.00. On top of that, if the customer made any calling card calls, there is a flat fee of $1.00.

In addition to these charges, Pa Bell also must collect various taxes and statutory user fees. Each customer, no matter how many or how few minutes they have used, must pay a Universal Subscriber Fee of $5.31. Each customer must pay a local tax of $1.25.

Each customer must also pay a state tax of 0.4% on the total amount of Pa Bell charges (but not including the USF or the local tax). Due to a recent revision in the tax code, there is now a ceiling of $0.20 on the state tax.

Mathematically the state tax will usually not be an exact number of cents; the result must be truncated to an integer number of cents, so a tax of 0.147 would be truncated to 0.14. (If you follow the advice on handling monetary amounts given later, this will happen automatically.)

Given the call data in the input file, you must calculate an itemized bill for the customer. The required format is shown in the sample output file given next.

Sample output:

Here is a sample output file, named `PhoneBill.txt`, which corresponds to the input data given above:

```
Programmer:  Bill McQuain
CS 1044 Project 4 Fall 2001

Customer:                                  John McBryde
Account:                                   540.231.2346
Billing date:                                02/24/2001

                                    Minutes    Charge
=====================================================
Peak:                                  2:10     13.00
Offpeak:                               5:10     11.30
Card:                                  1:32     23.00
=====================================================
Subtotal                                        47.30
Calling plan fee                                 4.95
Calling card fee                                 1.00
Universal connectivity charge                    5.31
Local tax                                        1.25
State tax                                        0.20
=====================================================
Total                                           60.01
```

As usual the first three lines just identify the programmer and project. The next three lines specify the customer, account and billing date. Next there is a blank line, and then the headers for the table with the bill amounts. The minutes and total charge for each call category are reported; there will be a line for each category even if there were no call minutes.

Next there is a subtotal for the per-minute charges, then the Pa Bell fees that were assessed. There will always be a calling plan fee. If there were no calling card calls then the line for the calling card fee is omitted.

Next the USF and local tax are reported, and then the state tax amount. (The state tax cannot be zero, so of course there will always be an entry for it.) Finally the grand total is reported.

If you have read the description of how the Curator scores your program in the *Student Guid*e, 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 are free to experiment with the horizontal layout but you should try to align the columns neatly.

Note that you must insert blank lines and divider lines as shown.

Some advice on handling monetary amounts:

The prices are decimal values and therefore pose some problems that we will discuss in class later. For now, you are advised 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.

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.
- Adopt a consistent indentation style and stick to it.

Your implementation must also 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.
- Note: you are explicitly forbidden to write any user-defined functions for this program. This will make the program somewhat repetitive, and physically longer than the alternative. To some extent, that's the reason for this restriction.

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 Guid*e), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guid*e.

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 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.