

**Putting the basics together:****Hokie Electric Company**

It's finally time to write a complete program. This project will use many of the C++ features that were illustrated in the source code you were given for the programming assignment as well as use some input and output basics that were discussed in class. The resulting program will read simple data for a Hokie Electric Company Customer and will produce an electric bill. This will require reading data from an input file, doing simple calculations, and writing a simple output file.

**Sample input data:**

Here is a sample input file, named `CustData.txt`, for the program. The first line specifies the name of the customer, and the second and third specify the customer's address. The fourth line contains the initial and final readings of the customer's electric meter. The meter readings will be integers and are specified in kilowatt-hours (kWh).

Here is an example input file:

```
Mike McBryde
303 Southgate Drive
Blacksburg, VA 24060
34567 35599
```

The input values are guaranteed to be syntactically and logically correct. The customer name and address information are just string data. All whitespace in the input file will be blanks.

**Calculations:**

In order to calculate the electric bill, you must first determine the Usage for this billing period. An electric bill consists of a number of smaller charges. Specifically there are charges for Generation, Fuel, Transmission, and Distribution. The formula for a given charge is in the following table:

```
Generation = Usage kWh * 1.778 cents/kWh
Fuel = Usage kWh * 1.463 cents/kWh
Transmission = Usage kWh * 0.419 cents/kWh
Distribution = $7.00 + Usage kWh * 1.576 cents/kWh
```

There is one important data representation issue in this project. As you will soon learn, computer hardware almost never stores decimal values exactly, and so any calculations done with decimal values are likely to be slightly wrong. To avoid that, your program is required to store all monetary amounts internally using integer variables. You may either store dollars

and cents separately, or store a total amount in cents. Failure to do this will result in a deduction when the TAs evaluate your implementation, and also may result in a deduction when the Curator scores your submission.

### Sample output:

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

```
Programmer: Rich Wheaton
CS 1044 Summer 2003 Project 2
-----

Mike McBryde
303 Southgate Drive
Blacksburg, VA 24060

Hokie Electric Company Current Charges
-----
Generation          18.34
Fuel                 15.09
Transmission         4.32
Distribution         23.26
-----
Total Due           61.01
```

The first two lines just identify the programmer and project. Next there is a separator line followed by a blank line. The next 3 lines specify the customer name and address. The next 3 lines are a blank line, the header shown in the sample, and another separator line. The next 4 lines contain the charges defined above and must be in the order shown in the sample. Next is a separator line, and finally is the Total, which is the sum of the 4 charges.

All monetary amounts are to be reported to two decimal places.

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 are free to experiment with the horizontal layout but you should try to align the columns neatly.

### Suggested implementation plan:

You should design your solution piece by piece. Begin by identifying the major tasks that have to be done, and then add detail for each of those tasks. Your implementation should be developed in the same manner. Here's a suggested order of implementation.

First, implement the input code to read the customer name and address, and write the output code to print them back out in the specified format. This will require declaring the appropriate stream variables and setting up the file streams, and declaring and using variables to store the input values. Test this code and make sure it produces correct results. Once you know this part works you should never have to worry about it again.

Second, implement the input code to read the electric meter readings. Add code to just echo the values you read in, to be sure you're reading them correctly.

Then add the code to calculate the charges and output them. Test this code. Now you should be ready to test all of the code.

By implementing the program feature by feature you let yourself concentrate on solving one small part of the problem at a time. You also should only have to test one part of it at a time as well.

**Evaluation:**

Everything that you have been told about testing in class 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 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* and submission link can be found at:

<http://www.cs.vt.edu/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.**