Fall 2020 CS 31

Programming Assignment 2
Say Cheese!

Time due: 11:00 PM Thursday, October 22

Before you ask a question about this specification, see if it has already been addressed in the spec or in the Project 2 FAQ. And read the FAQ before you turn in this project, to be sure you didn't misinterpret anything.

(Be sure you also do the homework accompanying this project.)

A consequence of Brexit is the United Kingdom's having to negotiate new trade deals with many countries. One sticking point in the US-UK trade talks is the U.S.'s proposed tariff on imports of British cheese. The job of writing a program that determines the proposed amount of import duty imposed on a particular quantity of cheese has come to you.

Your program must accept as input the type of cheese being imported (e.g., red windsor, cheshire, etc.), the value of the cheese, and the name of the importer. The output will tell how much import duty the importer will have to pay.

Here is an example of a dialog with the program (user input is shown here in **boldface**):

```
Cheese type: cheshire
Value: 3273
Importer: The Cheese Store of Beverly Hills
---
The import duty for The Cheese Store of Beverly Hills is $42.82
```

According to the import duty schedule:

- For the first 1000 dollars in value, the duty will be 1.1% of that value.
- In addition, for the next 12000 dollars in value (beyond the initial 1000 dollars), the duty will be increased by an additional amount of 2% of that additional value. However, if the cheese type is `cheshire` or `stilton` (so spelled, entirely in lower case), the amount of this additional duty is only 1.4% of that additional value instead of 2%. (One of the U.S. negotiators has some favorite cheeses and cut a special deal.)
- In addition, further duty will be imposed in the amount of 2.9% of the portion of the value that exceeds 13000 dollars.

As an example, The Cheese Store of Beverly Hills above would be assessed $11 for the first 1000 dollars in value, plus $31.82 for the next 2273 dollars in value (1.4% of 2273 instead of 2%, since the cheese type is cheshire), for a total of $42.82.

Here's another example:

```
Cheese type: caerphilly
Value: 404.25
Importer: Chucky's Cheese
---
The import duty for Chucky's Cheese is $4.45
```

You can test your understanding of the duty schedule by experimenting with this import duty calculator we found at the U.S. Trade Representative's web site.

Your program must collect the information for one importer in the manner indicated by the examples, and then write to `cout` a line with three hyphens only (no spaces or other characters), followed by exactly one line in a format required below. Our grading tool will judge the correctness of your program by examining only the line following the line with three hyphens (and verifying that there are no additional lines). That one line you write must be in one of the following four forms; the text must be **identical** to what is shown, except that italicized items are replaced as appropriate:

- If an empty string was provided for the cheese type:
  ```
  You must enter a cheese type
  ```
- If the value is not positive:
  ```
  The value must be positive
  ```
- If an empty string was provided for the importer:
  ```
  You must enter an importer
  ```
- If the input is valid and none of the preceding situations holds:
  ```
  The import duty for importer is $amount
  ```

In the last case, importer must be the importer the user entered, and amount must be the correct duty amount, shown as a number with at least one digit to the left and exactly two digits to the right of the decimal point. The lines you write must not start with any spaces. If you are not a good speller or typist, or if English is not your first language, be especially careful about duplicating the messages **exactly**. Here are some foolish mistakes that may cause you to get very few points for correctness on this project, no matter how much time you put into it, because the mistake will cause your program to fail most or all of the test cases we run:

- Not writing to `cout` a line with exactly three hyphens in all cases.
- Writing any spaces on the line that is supposed to have three hyphens.
- Writing more than one line after the line with three hyphens. Don't, for example, add a gratuitous "Blessed are the cheesemakers".
- Writing lines to `cerr` instead of `cout`.
- Writing lines like these:

```
The inport duty for Trader Joe's is $791.40      misspelling
The import Duty for Trader Joe's is $791.40      wrong capitalization
import duty for Trader Joe's is $791.40          missing text
The import duty for Trader Joe's will be $791.40  wrong text
The import duty for Trader Joe's is $ 791.40     extra space
The import duty for Trader Joe's is 791.40       missing dollar sign
The import duty for Trader Joe's is $791.40.     extra period
The import duty for Trader Joe's is $791.400     extra digit
The import duty for Trader Joe's is $791         missing decimal point and digits
```

Your program must gather the cheese type, the value, and the importer in in that order. However, if you detect an error in an item, you do not have to request or get the remaining items if you don't want to; just be sure you write to `cout` the line of three hyphens, the required message and nothing more after that. If instead you choose to gather all input first before checking for errors, and you detect more than one error, then after writing the line of three hyphens, write only the error message for the earliest erroneous item.

You will not write any loops in this program. This means that each time you run the program, it handles only one importer. It also means that in the case of bad input, you must not keep prompting the user until you get something acceptable; our grading tool will not recognize that you're doing that.

A string with no characters in it is the empty string. A string with at least one character in it is not the empty string, even if the only characters in it are things like spaces or tabs. Although realistically it would be silly to have an importer consisting of seventeen spaces and nothing more, treat that as you would any other non-empty string: as a valid importer. (Since you don't yet know how to check for that kind of situation anyway, we're not requiring you to.)

The one kind of input error that your program does **not** have to deal with, because you don't yet know how to do so, is not finding a number in the input where the value is expected. We promise that our grading tool will not, for example, supply the text `a whole lot` when your program requests the value. Notice that the value need not be an integer.

The correctness of your program must not depend on undefined program behavior. Your program could not, for example, assume anything about `n`'s value at the point indicated:

```
int main()
{
    int n;
    int m = 42 * n;  // n's value is undefined
    …
```

What you will turn in for this assignment is a zip file containing these three files and nothing more:

1. A text file named **duty.cpp** that contains the source code for your C++ program. Your source code should have helpful comments that tell the purpose of the major program segments and explain any tricky code.
2. A file named **report.docx** or **report.doc** (in Microsoft Word format) or **report.txt** (an ordinary text file) that contains:
    a. A brief description of notable obstacles you overcame. (In Project 1, for example, some people had the problem of figuring out how to work with more than one version of a program in Visual C++.)
    b. A list of the test data that could be used to thoroughly test your program, along with the reason for each test. You don't have to include the results of the tests, but you must note which test cases your program does not handle in the way this spec requires. (This could happen if you didn't have time to write a complete solution, or if you ran out of time while still debugging a supposedly complete solution.) For Project 1, for example, such a list, had it been required, might have started off like this:

        More voters than the total for Joe and Donald (1000, 413, 382)
        Fewer voters than the total for Joe and Donald (500, 413, 382)
        Zero total voters (0, 100, 100)
        Data giving a non-integer percentage (1000, 413, 382)
        More votes for Joe than Donald (1000, 413, 382)
        Equal votes for Joe and Donald (1000, 500, 500)
        ...
3. A file named **hw.docx** or **hw.doc** (in Microsoft Word format) or **hw.txt** (an ordinary text file) that contains your solution to the homework accompanying Project 2.

By October 21, there will be links on the class webpage that will enable you to turn in your zip file electronically. Turn in the file by the due time above. Give yourself enough time to be sure you can turn something in, because we will not accept excuses like "My computer died the afternoon of the day the assignment was due." There's a lot to be said for turning in a preliminary version of your program, report, and homework early (You can always overwrite it with a later submission). That way you have something submitted in case there's a problem later. Notice that most of the test data portion of your report can be written from the requirements in this specification, before you even start designing your program.

The writeup Some Things about Strings tells you what you need to know about strings for this project.

As you develop your program, periodically try it out under another compiler (g31 on cs31.seas.ucla.edu if you're doing your primary development using Visual C++ or Xcode). Sometimes one compiler will warn you about something that another is silent about, so you may be able to find and fix more errors sooner. If running your program under both environments with the same input gives you different results, your program is probably relying on undefined behavior (such as using the value of an uninitialized variable), which we prohibit.

If while working on this project you suddenly come over all peckish, consider stopping in at The Old Cheese Emporium.