Lottery Scraper

The goal of this assignment is to write a program that calculates the odds of profiting if you played several scratch-off lottery games. This project introduces writing output to a file.

Documentation and Style

This project **will have a portion of its grade devoted to documentation and code style.** Refer to the style guidelines on the course website for examples of what the TAs will be looking for.

This assignment is worth **100 points**. When you submit, Web-CAT will test that your program works correctly and creates the proper output, if your program works correctly you will receive **80 points** from the automated testing provided by Web-CAT. The remaining **20 points** are for style and documentation, which will be manually graded by the TAs, using the rubric posted on the course website.

Program Logic

Your program should start by asking the user to enter, at the console, the lowest amount that they would like to profit when playing the lottery. Read this value in as an integer. Your program should then prompt the user for an **output file name**, where a formatted table with our results will be written:

```
Enter the lowest dollar amount that you would like to profit:  10
Enter the output file name: output.txt
Generating report...
```

Once the minimum profit is known, you can compute the odds of profiting from each game. Your program should **read in data from an input file** (see the "Input File Format" section below): As your program reads each scratcher game, compute the odds and write them in the same order in a nicely formatted table (see the example) **to the output file**. This means you aren't required to use structures, sorting, or vectors, but they aren't prohibited.

Input File Format

Your program **must** read information about the scratcher games from a text file named `scratcher.txt`. Use of any other filename will cause the program to fail when it is graded on Web-CAT**.**

The file will consist of several multi-line "blocks", where each block represents a scratcher game. Each block will be formatted as follows:

- The first line will contain only the name of the game. The name may contain any kind of character – letters, digits, punctuation, or spaces. Hint: use `getline()`, you may need to use `ws` to consume additional whitespace before re-using get line.
- The second line contains two integers. The first is the **cost of the ticket** in dollars; the second is the **number of possible prizes that can be won** in that scratcher. The number of prizes has been given so that you can easily use a for-loop to read the following lines.

- After the second line, there is one line for each prize on the scratcher (in other words, if you call the second integer on the line above N, then there are N lines). Each prize line contains three integers. The first is the **value of the prize** in dollars; the second is the **total number of tickets** that were printed with that prize; the third is the number of tickets with that prize that have **not yet been claimed.**
- After the last prize line, there will be a blank line, and then a new block will start (or you've reached the end of the file).

Example Input and Output
Below is an example input file with three games. You can use this during your own development.

```
SMALL BEANS
1 5
10000 10    10
5000  50    8
100   100   6
10    1000  4
1     10000 2

PIRATE'S BOOTY, ARRR
10 6
50000 20    0
10000 100   0
1000  500   0
100   2000  738
10    7500  2945
1     10000 4476

BIG MONEY HU$TLA$
20 7
1000000 10    7
500000  50    29
10000   100   78
1000    500   396
100     2000  1439
20      5000  3218
10      10000 6210
```

The first game is called "SMALL BEANS", a $1 game with 5 possible prizes: $10,000, $5,000, $100, $10, and $1. For the $10,000 prize, 10 tickets were printed and 10 of them have not yet been claimed; for the $5,000 prize, 50 tickets were printed and there are 8 left; and so on. The other two games are designed similarly.

Your program should start by prompting the user to enter a dollar amount that is the smallest amount that they want to profit (meaning that the prize must be larger than the **profit plus the cost of the ticket**; more on this later).

Once you know the desired profit (and prompt for the output file name), you can read the games from the input file. For each game, you should print a neatly formatted line of output to the file that includes the following information:

- The name of the game, exactly as it was read from the input file.
- The cost of a ticket for that game, preceded with a dollar sign (you may have spaces between the dollar sign and the amount, for alignment purposes).
- The odds of winning your desired profit in that game, written as "1 in N", where N will be computed as described below. The denominator should be printed to at least two decimal places. If it is not possible to profit from that game, print "Not possible" instead of the odds.

To compute the odds, you need to look at the information for each of the prizes on that scratcher. First, you need to compute a running sum of total number of remaining tickets for that game, regardless of prize value. Then, you also need to compute a running sum of the total number of remaining tickets that would result in a profit; in other words, the number of remaining tickets where the prize value is greater than the desired profit plus the cost of the ticket. (We're not interested in breaking even, so make sure you check strictly greater-than.)

Then, we want the odds to be "1 in something", so we compute the "something" by dividing the total number of remaining tickets by the number of remaining profitable tickets. (Notice that this is the opposite of what we would do if we were computing the probability as a percentage. Also note that we must take special care that we handle the "Not possible" case correctly, since that would involve dividing by zero.)

For the example input file above, imagine that the user said that they wanted to see what their odds were of profiting $1,000 on any of those games. **Your output file** should look something like this:

```
Game                                        Cost  Odds
------------------------------------------------------------
SMALL BEANS                                 $ 1  1 in 1.67
PIRATE'S BOOTY, ARRR                        $10  Not possible
BIG MONEY HU$TLA$                           $20  1 in 99.80
```

The spacing does not have to match up exactly, but it should look reasonably close to this. I don't want to define the output format too rigidly, so **give yourself plenty of time** to work on it in case you run into last minute problems.

What to Submit
For this assignment you should submit your `main.cpp` file from inside of your project directory.

This assignment will be graded automatically. Test your programs thoroughly before submitting them. Make sure that your programs produce correct results for every logically valid test case you can think of. Do not waste submissions on untested code, or on code that does not compile with the supplied code from the course website.

Web-CAT will assign a score based on runtime testing of your submission; your best score will be counted; the TAs will later verify that your best submission meets the stated restrictions, and assess penalties if not.

To submit this assignment:

1. Visit http://web-cat.cs.vt.edu in your web browser.
2. Enter your Virginia Tech PID and password in the appropriate fields on the log-in screen, and make sure that **Virginia Tech** is selected as the institution. Click **Login**.
3. The Web-CAT home screen will display useful announcements and assignments that are currently accepting submissions. Find the assignment that you want to submit in the table, and click the "Submit" button next to it.
4. Click the **Browse...** button and select the file you want to upload. The homework assignments and programming projects for this course should be self-contained in a single **main.cpp** file, so you can simply select that one file.
5. Click the **Upload Submission** button. The next page will ask you to review your selection to ensure that you have chosen the right file. If everything looks correct, click **Confirm**.

The next page will show that your assignment is currently queued for grading, with an estimated wait time. This page will refresh itself automatically, and when grading is complete you will be taken to a page with your results.

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 submitted file:

```
//    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 an allowed source, such as a text book or course
//      notes, that has been clearly noted with a proper citation in
//      the comments of my program.
//
//    <Student Name>
```

Failure to include this pledge in a submission will result in the submission being disallowed during code review.