One Born Every Minute

CS260

Contents:

Please read the instructions on How to Prepare and Submit Assignments.

This assignment will give you an opportunity to demonstrate your ability to work with computer arithmetic, basic control flow and interactive I/O operations.

1 Instructions

1. Read the problem description below.
2. Get the files for this assignment. Compile the program "as is" to familiarize yourself with it. You will receive error messages. Read them. They are hints as to what is missing from the current code.
3. Your focus in this assignment will on the implementation of the `solve` function in `lottery.cpp`.
4. Your code will be evaluated both on its ability to pass the tests provided.

> You are not limited to using my system tests. It's always a good ide ato run the program on data of your own choosing as well.

2 Assignment Management

1. Get the starting code for this assignment. (Right click and "save … as" to do a single download of the entire zip file).
2. Submit your code.
3. View your grade report..

It may take a few minutes after you submit before your grade report is ready for viewing.

- If it takes more than an hour, contact your instructor.

2.1 Files that you may change:

3 Assignment management

3.1 Actions

1. Get the starting code for this project and register your GitHub repository.
2. View your current grade report..
    - A request for a *preliminary* grade report will be automatically filed each time you push changes to your repository.

      It may be several minutes before grading begins, depending on how many requests are in front of yours. Grading itself can take anywhere from a few seconds to many minutes.
    - On rare occasions the request from GitHub to our servers may fail. Also, your web browser may have cached the old report and be showing that to you.

      If more than an hour has passed since you last pushed changes, and your grade report has not updated, you should:

        A. Clear your browser's cache (files only, not cookies or history) and then try to view the grade report again.
        B. If that does not work, request preliminary grading of your most recent push.

3.2 Files that you may change:

- `lottery.cpp`

4 Problem Description

4.1 Lottery Odds

Playing the state lottery can be fun, even if it is bit of a sucker bet. In this assignment, you will be writing a program to compute odds of winning the lottery. We will be modeling a basic lottery in which you choose some number of numbers, all of which are treated identically, without the bells and whistles such as ``power balls" or other rules that treat one selected number as different from any of the others.

Let us say that a lottery is based on a list of N different numbers and that, when you buy a ticket, you select k of those. In a basic lottery, the order in which the numbers are drawn makes no difference. So if you have a ticket on which you have selected "1", "4", and "6", you win if the state draws the numbers "1 4 6" or "4 6 1" or 6 4 1" or … The possible selections that, as in this example, ignore the ordering of the selections are called *combinations*.

The number of combinations of k things chosen from among N possibilities comes up in a lot of mathematical problems and is denoted as $\binom{N}{k}$. The usual formula given for this is

$$\binom{N}{k} = \frac{N!}{k!(N-k)!}$$

where ! is the *factorial* operator (i.e., $k! = 1 * 2 * \ldots * k$).

The problem with that formula is that the value $N!$ gets so large for the kinds of numbers that appear in real lotteries that it cannot be stored in a C++ integer and can only be stored in a floating point number (`double`) at the cost of losing many, many digits of precision. For example, 42! is 405006117752879898543142606244511569936384000000000. Luckily, there is another way to compute this which is mathematically equivalent but better suited to computation:

$$\binom{N}{k} = \prod_{i=1}^{k} \frac{N-i+1}{i}$$

The $\prod_{i=1}^{k}$ notation indicates the product obtained by multiplying all the terms that can be obtained by replacing i by all the integers from 1 to k, so this is a compact way of saying

$$\binom{N}{k} = \frac{N-1+1}{1} * \frac{N-2+1}{2} * \ldots * \frac{N-k+1}{k}$$

Write a program that prompts the user to enter the number of numbers in the lottery and the number of those selected on a ticket and that then uses the above formula to compute the odds of that ticket winning the lottery.

4.2 Input and Interaction

Input to the program will be taken from standard input (`cin`).

Your program should prompt the user with the phrase

```
How many numbers are printed on the lottery ticket?
```

with a blank space (but no line feed) after the question mark. It should read in the answer (as an integer).

Then your program should prompt the user with the phrase

```
How many numbers are selected in the lottery drawing?
```

again with a blank space (but no line feed) after the question mark. It should read in the answer (as an integer).

4.3 Output

Output is written to the standard output (`cout`).

After the interactive prompts described above, the program should print one empty line and then print a line containing the appropriate answer, as follows.

If either of the input numbers is zero or negative, of if the number of numbers to be selected is greater than the number of printed numbers, the input makes no sense. Your program should print

```
This is not a possible lottery.
```

and then immediately terminate.

If the numbers are sensible, then compute the number of possible lottery draws as described above and print

`Your chances of winning are 1 in` *NNN*

where *NNN* is replaced by the appropriate integer number.

4.4 Example

```
How many numbers are printed on the lottery ticket? 42
How many numbers are selected in the lottery drawing? 6

Your chances of winning are 1 in 5245786
```

User inputs are highlighted.

5 Notes

1. It is important to note that the output format (including the text of the prompts) be followed exactly.

2. In this assignment and in all assignments in this course, when we talk of printing a "line" or reading a "line", we refer to a properly terminated line – a string of characters followed by a newline `"\n"` or `endl`.

   Note, however, that when prompting for input, we do not usually terminate the prompt with `"\n"` or `endl`, because these would force the human's response onto a separate line, which often looks ugly. Instead we use `flush`:

   ```
   cout << "Give me some input, please: " << flush;
   cin >> stuff;
   ```

   resulting in a screen that looks like:

   ```
   Give me some input, please: 42
   ```

   with the typed response appearing immediately after the prompt.

   Other than this special case of prompting for input, you should *always* terminate your output lines with `endl` or `"\n"`.

3. In this and in all assignments, do not rely on the single testS & examples that I show in the assignment. You should always develop your own tests. For example, you might want to start with some test cases that are simple enough for you to compute the proper answer "by hand". For example, lotteries in which you select 1 of 5 numbers, 2 of 5 numbers, 3 of 5 numbers, 4 of 5 numbers, and 5 of 5 numbers. (You may notice an interesting pattern to the answers.)

   For more complicated tests, if you are uncertain what the appropriate output should be, note that I provide you with compiled versions of my own solution (for Windows and for our Unix machines), so you can run your own tests through it.

4. The product of fractions formula for $\binom{N}{k}$ will work properly only if you do the computations using floating point numbers. (Use `double` rather than `float` to get the best precision possible.)

   However, you need to convert the final answer to an integer before printing it. If you simply assign the answer to an integer variable, any fractional part (after the decimal point) will be truncated. Now, ideally, there would not *be* anything after the decimal point, but floating point calculations are always subject to round-off error. So the safe thing to do is to round the floating point answer to the closest integer. This is done by adding 0.5 to the number, then assigning it to an integer:

   ```
   long intVariable = (long)(0.5 + doubleVariable);
   ```

   so that, if the double variable has a fractional part that is larger than 0.5, adding the additional 0.5 pushes it over to the next integer before you truncate.

- In this and all assignments in the course, you should be setting your compiler/IDE to the C++17 standard. (Most IDEs and compilers default to the much older C++98 standard.) If you are using the provided `makefile`, this will be handled automatically. If you do need to configure your IDE for this, instructions are in the FAQ.