CSci 1113: Introduction to C/C++ Programming for Scientists and Engineers
Homework 2
Spring 2018

Due Date: Thursday, Feb. 15, 2018 before 11:55pm.

Instructions: This is an individual homework assignment. There are two problems worth 20 points each. Solve the problem below by yourself (unlike the labs, where you work collaboratively), and submit the solution as a C++ source code file. Here are a few more important details:

1. Unlike the computer lab exercises, this is not a collaborative assignment.

2. Because all homework assignments are submitted and tested electronically, the following are important:
    - You follow any naming conventions mentioned in the homework instructions.
    - You submit the correct file(s) through Moodle by the due deadline.
    - You follow the example input and output formats exactly given in each problem description.
    - **Regardless of how or where you develop your solutions, your programs compile and execute on cselabs computers running the Linux operating system.**

3. You should test your program on other test cases (that you make up) as well. Making up good test cases is a valuable programming skill, and is part of ensuring your code solution is correct.

Problem A: Triangle (20 points)
Draw a triangle (like shown below) with dots in the box where the triangle is not and plus symbols where the triangle is. Ask the user the size of the triangle (you can assume the number entered will be odd).

Note: the height of the triangle is **not** the variable that the user entered.

Example 1 (user input is underlined):
```
Triangle size?
1
+
```

Example 2 (user input is underlined):
```
Triangle size?
3
.+.
+++
```

Example 3 (user input is underlined):

```
Triangle size?
13
......+......
.....+++.....
....+++++....
...+++++++...
..+++++++++..
.+++++++++++.
+++++++++++++
```

When you are done, name the source code file <username>_2A.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_2A.cpp. Then submit your program using the HW 2 Problem A submission link in Moodle.

Problem B: Upgrading gear (20 points)
Suppose you are playing an MMO where you can upgrade your weapons and armor by spending some money. However, the system for upgrading involves both randomness and is a bit convoluted (it is not easy to mathematically find the answer to this problem... but possible).

The system is as follows:

- There is a base chance of 20% for a successful equipment upgrade.
- If you fail to upgrade, however, it will cost you some money to be able to try again.
- To make things more complicated, each time you fail, you get a progressively higher chance of success on the next try (5% increase per each consecutive fail).

For example, let the cost to upgrade be 700 coins. The first time you try to upgrade you have a 20% chance. If you fail to upgrade you have to spend 700 coins, but now you have a 25% chance of successfully upgrading. If you fail a second time, you would pay an additional 700 coins (total of 1,400 now) but your chance of success is now 30%. If you succeed in upgrading, you do not need to pay anything.

Some equipment costs more or less to fix. You can take advantage of this system by building up failures on cheap equipment, then once you have a few successive failures you can try upgrading more expensive equipment (that you actually want to upgrade).

Write a program to figure out how many successive fails you should receive on cheap equipment before trying to upgrade more expensive equipment. For this program you can assume, *with cheap equipment*, the cost of "x" cumulative fails is:

$$500 \cdot x \cdot 1.05^{x}$$

What your program needs to do:

1. The **retry cost**: Ask the user how much it costs to try again on the expensive equipment.
2. The simulation: To figure out what the best strategy is, use a simulation of one million different tries at making upgrades. For each try:
    - Start with a fixed “x” value. Let the equation above determine the cost of failing to upgrade “x” times.
    - Keep attempting to upgrade again (and again) until you succeed.
    - Keep track of the costs that you are accumulating.
3. The comparison: Calculate the average costs associated with each “x” value.
4. The conclusion: Compare these averages and determine the most effective “x” value. Remember:
    1. “getting 0 successive failures” means starting immediately with the more expensive equipment (i.e. starting with f(0) = 0 as in the first example in red below)
    2. “getting 16 successive failures” will give you a 100% chance to succeed on the first try (but is quite expensive to get).
5. Output the **minimum average cost** and the **corresponding x**

For the best strategy, show both the average cost (including how much it takes to get the original successive failures) to upgrade and how many successive failures to get before starting with the expensive equipment.

Note: with one million trials, it will take your computer a bit to compute this (a few seconds). Also since these are random, the numbers won't be exact, but the first couple digits should be the same.

Note2: I have added “zeroToOne.cpp” to show how to generate a random number between zero and one (though this is pretty easily google-able).

As some people are still confused, I will give a better example. Let f(x) = 500*x*1.05^x, which is the formula above. Here “x” represents how many consecutive fails you start out with and f(x) the cost. So, if you started trying to upgrade immediately, you would pay:

- Success on first attempt (20% chance to succeed) → pay = 0
- Success on second attempt (25% chance to succeed) → pay = 1 * *retry cost*
- Success on third attempt (30% chance to succeed) → pay = 2 * *retry cost*
- Success on fourth attempt (35% chance to succeed) → pay = 3 * *retry cost*

If the retry cost is high, it makes sense to build up cumulative failures using f(x) instead. Suppose you wanted to start with 3 cumulative failures to upgrade cheap equipment (f(3) = 1736.44). Then you would have the following simulation: (note the starting at 35% chance since we’ve failed 3 times)

- Success on first attempt (35% chance to succeed) → pay = 1736.44
- Success on second attempt (40% chance to succeed) → pay = 1736.44 + (1 * *retry cost)*
- Success on third attempt (45% chance to succeed) → pay = 1736.44 + (2 * *retry cost)*
- Success on fourth attempt (50% chance to succeed) → pay = 1736.44 + (3 * *retry cost)*

TL;DR: First you need to average the cost if you start with x=0 (and 20% initial success), then average another million trials with x=1 (start at 25% success), then x=2... all the way to x=16. You then need to determine which of these x values produced the minimum average. (The first number output is the average, the second is x value associated with the minimum average.)

Example 1, fixed (user input is underlined):

```
What is retry cost?
2000
Average cost:
4517.44
Should get successive failures:
2
```

Example 2 (user input is underlined):

```
What is retry cost?
100
Average cost:
242.668
Should get successive failures:
0
```

When you are done, name the source code file <username>_2B.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_2B.cpp. Then submit your program using the HW 2 Problem B submission link in Moodle.

Problem C: Triangle of Power (5 points, extra credit)
Modify your code from part A to make a "triforce" (each part of the triforce should be the triangles you made from part A).

Example 1 (user input is underlined):

```
Triangle size?
1
.+.
+.+
```

Example 2 (user input is underlined):

```
Triangle size?
3
...+...
..+++..
.+...+.
+++.+++
```

Example 3 (user input is underlined):

```
Triangle size?
13
.............+.............
............+++............
...........+++++...........
..........+++++++..........
.........+++++++++.........
........+++++++++++........
.......+++++++++++++.......
......+.............+......
.....+++...........+++.....
....+++++.........+++++....
...+++++++.......+++++++...
..+++++++++.....+++++++++..
.+++++++++++...+++++++++++.
+++++++++++++.+++++++++++++
```

When you are done, name the source code file <username>_2C.cpp. Here you replace <username> with your U of M email address; for example, if your email address is smithx1234@umn.edu, your file should be named smithx1234_2C.cpp. Then submit your program using the HW 2 Problem C submission link in Moodle.