CSci 1113: Introduction to C/C++
Programming for Scientists and Engineers
Homework 8
Spring 2018

Due Date: Thursday, April 19, 2018 before 11:55pm.

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

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 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.

In Workout 1 for Lab 9 you made a polynomial class to add together two degree 2 polynomials. Problem A is similar, so while this problem will not build off this lab assignment. but it might be useful to reference.

Problem A: Polynomial multiplication (20 points)

For this problem, you must write code that can multiply polynomials together (i.e. FOIL at a arbitrary high dimension). First you should prompt the user for the number of coefficients and then read each coefficient (starting from the constant term). Repeat this a second time for a second polynomial. **You must use dynamic memory to assign these polynomials. You should also delete all dynamically created variables before the end of your program.** (You **do not** need to use a class to store these polynomials as we used in Lab 9 Workout 1.)

Once you have these two polynomials, find the product of the two polynomials. (Hint: It might be easiest to make a third dynamic array.) **Ensure that your formatting matches the examples below.** You can assume that the coefficients of the polynomials are all integers. You can also assume the polynomials have at least one coefficient.

```
Example 1 (user input is underlined):
How many coefficients are in the first polynomial? 3
What are the coefficients (lowest power first)? 3 2 1
How many coefficients are in the second polynomial? 3
What are the coefficients (lowest power first)? 1 2 1
(3) + (2)x^1 + (1)x^2
times
(1) + (2)x^1 + (1)x^2
-----
(3) + (8)x^1 + (8)x^2 + (4)x^3 + (1)x^4
```

```
Example 2 (user input is underlined):
How many coefficients are in the first polynomial? 3
What are the coefficients (lowest power first)? 1 -2 3
How many coefficients are in the second polynomial? 3
What are the coefficients (lowest power first)? -3 -2 1
(1) + (-2)x^1 + (3)x^2
times
(-3) + (-2)x^1 + (1)x^2
-----
(-3) + (4)x^1 + (-4)x^2 + (-8)x^3 + (3)x^4

Example 3 (user input is underlined):
How many coefficients are in the first polynomial? 2
What are the coefficients (lowest power first)? 1 2
How many coefficients are in the second polynomial? 2
What are the coefficients (lowest power first)? 3 4
(1) + (2)x^1
times
(3) + (4)x^1
-----
(3) + (10)x^1 + (8)x^2

Example 4 (user input is underlined):
How many coefficients are in the first polynomial? 1
What are the coefficients (lowest power first)? 2
How many coefficients are in the second polynomial? 5
What are the coefficients (lowest power first)? 5 -4 0 -2 1
(2)
times
(5) + (-4)x^1 + (0)x^2 + (-2)x^3 + (1)x^4
-----
(10) + (-8)x^1 + (0)x^2 + (-4)x^3 + (2)x^4
```

When you are done, name the source code file <username>_8A.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_8A.cpp. Then submit your program using the Homework_8A submission link in Moodle. It is important that you follow the file naming conventions very carefully. For example, note your username should be all in lowercase, you should not include “@umn.edu”, the file name should contain an underscore (not a dash), the ‘A’ in the “8A” part is upper case, the extension is .cpp, etc. Following rigorous naming conventions is something computer programmers often must do in “real life” programming, and so submitting your program with the correct name is part of doing this assignment correctly.

Problem B: Dice rolling class (20 points)

Start by downloading the "dice.cpp" file from the webpage. This will come with a small main() and a function to parse the user input into an array of ranges for dice. The format expected for the user to input is either "xdy", where x and y and integers or "+x" where x is an integer. For example: "2d6+1" will represent rolling two 6-sided dice, summing the faces then adding one to the result. Multiple additions or dice can be incorporated by spacing, for example: "2 d 4 + 1 3d6 +4" will be two 4-sided dice, three 6-sided dice plus 5. (Note: a "die" always rolls 1 to the maximum number. So "1d20" or one twenty-sided die will be a random number between 1 to 20.)

Your program must have the following:

(1) Create a Dice class that has:

1. Two private integer variables to store the minimum and maximum roll possible.
2. Two constructors (a default and a nondefault one that takes two integers for min/max values of rolls)
3. Overload the class with cout to display a random roll of the dice. In your output, you should include the following line in main():

```
cout << "Sample [1,6] roll: " << Dice(1,6) << endl;
```

... this should output something like:

```
Sample [1,6] roll: 3
```

(2) Create a roll() function that returns a random number uniformly distributed between the minimum and maximum dice roll value.

(3) Create a dynamic array of the Dice class created in (1), and use this to compute the mean, minimum, and maximum as shown below. (And properly delete the dynamic array!)

After the single getline() in the sample code, your program should ask the user how many rounds they wish to roll. You may assume that the user does not enter a malicious input. Then roll all the dice this number of times and display the following: (in the **format and order** shown in the examples)

1. The minimum roll seen.
2. The maximum roll seen.
3. The average of all the rolls (I did not change the default cout settings for this).

```
Example 1 (user input is underlined, this is two four-sided dice):
What do you want to roll?  2d4
How many rounds do you want to roll? 2
Minimum roll: 4
Maximum roll: 5
Average roll: 4.5
```

```
Example 2 (user input is underlined, this is two four-sided dice):
What do you want to roll?  2d4
How many rounds do you want to roll? 2
Minimum roll: 6
Maximum roll: 7
Average roll: 6.5

Example 3 (user input is underlined, one six-sided die plus 2):
What do you want to roll?  1d6+2
How many rounds do you want to roll? 1000
Minimum roll: 3
Maximum roll: 8
Average roll: 5.517

Example 4 (user input is underlined, one one-thousand-sided die (i.e.
1 to 1000)):
What do you want to roll?  1d1000
How many rounds do you want to roll? 1
Minimum roll: 918
Maximum roll: 918
Average roll: 918

Example 5 (user input is underlined, this is one six-sided die, four
seven-sided dice, one twenty-sided die and plus ten):
What do you want to roll?  1  d 6   +   2  4d7+8  1d20
How many rounds do you want to roll? 1000000
Minimum roll: 16
Maximum roll: 64
Average roll: 39.9987
```

When you are done, name the source code file <username>_8B.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_8B.cpp. Then submit your program using the Homework_8B submission link in Moodle. It is important that you follow the file naming conventions very carefully. For example, note your username should be all in lowercase, you should not include “@umn.edu”, the file name should contain an underscore (not a dash), the ‘B’ in the “8B” part is upper case, the extension is .cpp, etc. Following rigorous naming conventions is something computer programmers often must do in “real life” programming, and so submitting your program with the correct name is part of doing this assignment correctly.