NYU, Tandon School of Engineering
Extended Bridge to CS — Summer 2021

Homework #5
Due by Friday 8/13 11:59pm

Submission instructions:

1. For this assignment, you should turn in 3 files:
   - Two '.cpp' files, one for each question 1 and 2.
     Name your files 'YourNetID_hw5_q1.cpp', and 'YourNetID_hw5_q2.cpp'.
   - A '.pdf' file with your answers for questions 3-5.
     Each question should start on a new page!
     Name your file 'YourNetID_hw5_q3to5.pdf'

2. You must type all your solutions. We will take off points for submissions that are handwritten.

3. **You should submit your homework in the Gradescope system.**

4. **You can work and submit in groups of up to 4 people. If submitting as a group, make sure to associate all group members to the submission on gradescope.**

5. Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose the most appropriate control flow statements, etc.

6. For the math questions, you are expected to justify all your answers, not just to give the final answer (unless explicitly asked to).
   As a rule of thumb, for questions taken from zyBooks, the format of your answers, should be like the format demonstrated in the sample solutions we exposed.

Question 1:

Write a program that reads a positive integer `n` from the user and prints out a `nxn` multiplication table. The columns should be spaced by a tab.

Your program should interact with the user **exactly** as it shows in the following example:

```
Please enter a positive integer:
10
1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100
```

Question 2:

Implement a number guessing game. The program should randomly choose an integer between 1 and 100 (inclusive), and have the user try to guess that number.

Implementations guidelines:

1. The user can guess at most 5 times.
2. Before each guess the program announces:
    - An updated guessing-range, taking in to account previous guesses and responses.
    - The number of guesses that the user has left.
3. If the user guessed correctly, the program should announce that, and also tell how many guesses the user used.
4. If the user guessed wrong, and there are still guesses left, the program should tell the user if the number (it chose) is bigger or smaller than the number that the user guessed.
5. If the user didn't guess the number in all of the 5 tries, the program should reveal the number it chose.
6. Follow the execution examples below for the exact format.

In order to generate random numbers (of type **int**), you should first call the **srand** function (one time) to initialize a seed for the random number generator. Then, you can call the **rand** function repeatedly to generate random integers.

Follow the syntax as demonstrated in the code below:

```
1.  #include <iostream>
2.  #include <cstdlib>
3.  #include <ctime>
4.  using namespace std;
5.
6.  int main() {
7.      int x1, x2, x3, x4;
8.
9.      srand(time(0));
10.
11.     x1 = rand();
12.     x2 = rand();
13.     x3 = rand() % 100;
14.     x4 = (rand() % 100) + 1;
15.
16.     cout<<x1<<" "<<x2<<" "<<x3<<" "<<x4<<endl;
17.
18.     return 0;
19. }
```

Note that we first included the **cstdlib** and **ctime** libraries (lines 2 and 3).
In line 9 we called **srand(time(0))** to create the seed for the random number generator. Then, in lines 11-14, we created 4 random numbers: The first two were taken from the entire positive range of **int** variables. For **x3** and **x4** we used arithmetic manipulation to change the range. **x3** would be in the range 0-99 (since these are the possible remainders when dividing a number by 100), and **x4** would be in the range of 1-100 (shifting the range 0-99 by 1).
Further reading regarding random number generation can be found in the textbook on pages 188-189.

Your program should interact with the user **exactly** as it shows in the following two examples:

Execution example 1:

I thought of a number between 1 and 100! Try to guess it.
Range: [1, 100], Number of guesses left: 5
Your guess: 15
Wrong! My number is bigger.

Range: [16, 100], Number of guesses left: 4
Your guess: 34
Wrong! My number is smaller.

Range: [16, 33], Number of guesses left: 3
Your guess: 23
Congrats! You guessed my number in 3 guesses.

Execution example 2:

I thought of a number between 1 and 100! Try to guess it.
Range: [1, 100], Number of guesses left: 5
Your guess: 15
Wrong! My number is bigger.

Range: [16, 100], Number of guesses left: 4
Your guess: 50
Wrong! My number is smaller.

Range: [16, 49], Number of guesses left: 3
Your guess: 3
Wrong! My number is bigger.

Range: [16, 49], Number of guesses left: 2
Your guess: 34
Wrong! My number is smaller.

Range: [16, 33], Number of guesses left: 1
Your guess: 20
Out of guesses! My number is 25