# CIS7 Unit 6 Lab: Programming Probability

In this lab exercise, we will apply the learned concepts in Chapter 9, probability, to C++ programming. Please review additional lab instruction content and complete as instructed in the procedures.

## Generating Random Number

**srand()** is used in C++ to help in the generation of random numbers by **seeding rand fucntion with a starting value**. Usually random numbers are calculated by taking the previous number (or the seed) and then do many operations on that number to generate the next.

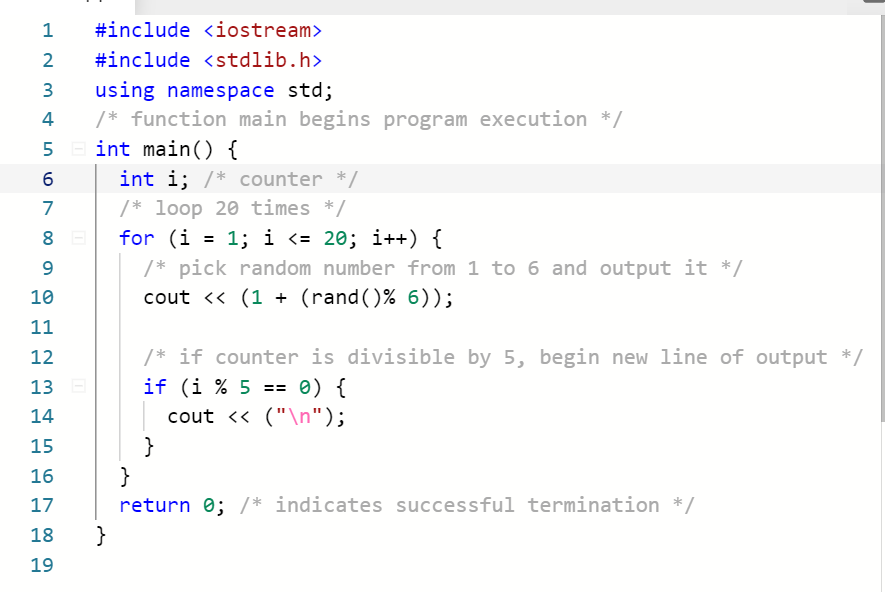
In order to generate random-like numbers, **srand()** is usually initialized to some distinctive runtime value, like the value returned by function time (declared in header **<ctime>.**

### Rolling Dice:

To simulate **20 rolls of a six-sided die and print the value of each roll**. The function prototype for function rand is in **<stdlib.h>**. We use the **remainder operator (%)** with rand() % 6 to produce **integers from 0 to 5**, called **scaling the number** (6 is called the **scaling factor**):

We then shift the range of numbers produced by adding 1 to our previous result. The output of these numbers differ from one compiler to another as it is supposed to be random.

**Example 1:** Using rand() to generate 20 rolls of a dice.



1. Input Example 1 program in IDE and run the program. Take a screen capture and answer the following questions:
2. Calculate the probability of obtaining a 1 using a six-face dice in 20 rolls.
3. Calculate the probability of obtaining a number other than 1 using a six-face dice in 20 rolls.
4. Review the program output how many times does each number scale/appear?
5. Use result from 1C, determine the probability each number appearing in the 20 rolls.
6. Which number appears the most frequent in the output?
7. Rerun the program. Is the result the same or different? Explain should the result be the same or different.
8. Assuming that there are 1 through 8 numbers to be randomly selected, replace line 9 with (% 8). Rerun the program.
9. Take a screen capture, with updated value in the program.
10. Provide the frequency of numbers generated in the program for 20 rolls.
11. Which number appears the most in the 20 rolls? Is the outcome different from 6 face dice?
12. Increase the number of rolls to 40 (change 20 to 40) and rerun the program. Which number appears most frequent in 40 rolls?

### Die Face Frequency

**Example 2:** To show that these numbers occur approximately with equal likelihood, let’s simulate 6000 rolls of a die with the program above so we should say that each number from 1 to 6 should appear approximately 1000 times.

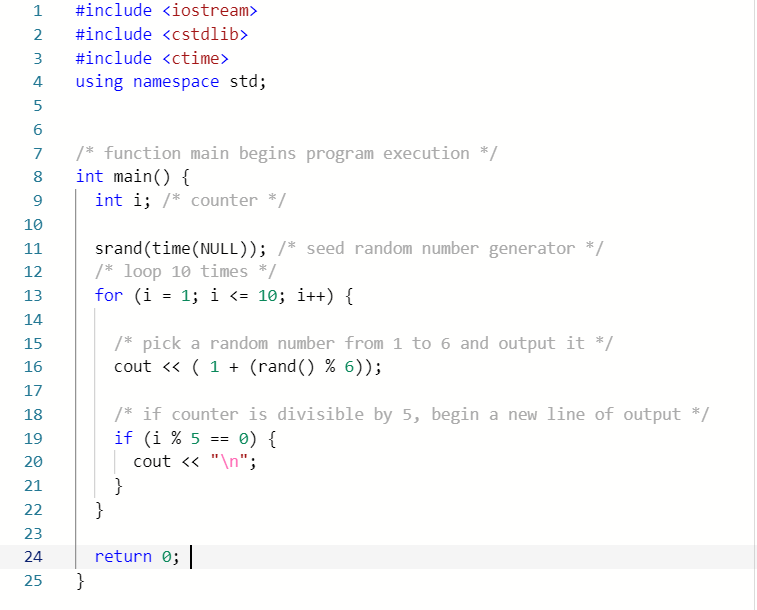


1. Input the Example 2 program into IDE, run the program and take screen capture.
2. Run the program a few times. Does it provide different result each time?
3. Which face has the highest frequency? What is its probability in 6000 rolls?
4. Which face has the lowest frequency? What is its probability in 6000 rolls?
5. Edit the program to generate results for 100 rolls. Which face has the highest and the lowest frequency? What are the probabilities of these faces in 100 rolls?
6. Edit the program to generate result for 1000 rolls. Which as has the highest and the lowest frequency? What are the probability of these faces?
7. Evaluate the results from Exercise 1 and Exercise 2. Describe the similarities or differences in the results based on the use of rand().What general observation can you draw about rand() from Exercise 1 and Exercise 2?

### SRAND

In the previous exercise, Exercise 2, rand() generates pseudorandom numbers. Calling **rand()** repeatedly produces a sequence of numbers that appears to be random. However, the sequence repeats itself each time the program is executed this can help you debug your program that uses **rand** function. Once a program has been thoroughly debugged, it can be conditioned to produce a different sequence of random numbers for each execution. This is called randomizing and is and that can be done using the standard library function **srand.** Function **srand** takes an unsigned integer as a parameter and seeds function rand to produce a different sequence of random numbers for each execution of the program.

**Example 3**: Using srand() to generate 10 rolls for six-face dice.



**srand( time( NULL ) );** causes the computer to read its clock to obtain the value for the seed automatically. Function time returns the number of seconds that have passed since midnight on January 1, 1970. This value is converted to an unsigned integer and used as the seed to the random number generator. Function time takes NULL as an argument and it is found in the header**< ctime>**

1. In this program, we will explore SRAND function(). Input program in IDE and run the program. Provide screen capture.
2. Run the program 3 times. Are the results the same each time the program is ran?
3. Calculate the probability of face 6 in 10 rolls.
4. Note the number with the highest frequency. What is its probability out of the 10 rolls?
5. Note the number with the lowest frequency. What is its probability out of the 10 rolls?
6. Increase the number of rolls to 20, line 11, and rerun the program. Is the output different from #3A?
7. What is the number with the highest frequency? What is its probability for 20 rolls?
8. What is the number with the lowest frequency? What is its probability for 20 rolls?
9. Compare result between 10 rolls and 20 rolls. Note and explain the differences or similarities.
10. What can you conclude about function **rand() and srand()?**

Login Canvas and submit document (.docx or .pdf) containing lab screen captures and answers.