

# CS 103 Lab - Arrays and Functions

---

## 1 Introduction

In this lab you will write a program to plot out a simple version of the “bell curve” or “normal distribution” based on the sum of 4 random dice rolls.

## 2 What you will learn

After completing this lab you should be able to:

- Prototype and write the implementation of a function
- Understand array indexing
- Understand how to use loops to iterate over arrays
- Introduce randomness into your program

## 3 Background Information and Notes

The “normal distribution” is a very important family of probability distributions. If you take any random event that produces numerical values, and repeat it often, the average of those values, although random, behaves in a predictable way. In this lab we are using only a small number of rolls (4), which is not infinite, but enough to yield a curve that resembles the bell curve. Here is a sample run of your program:

```
How many rolls do you want? 500 (this is the user input)
4:
5:X
6:XXX
7:XXXXXXXXXXXXXXXXX
8:XXXXXXXXXXXXXXXXX
9:XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
10:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
11:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
12:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
13:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
15:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
16:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
17:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
18:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
19:XXXXXXXXXXXXXXXXXXXXX
20:XXXXXXXXXX
21:XXX
22:XX
23:
24:X
```

In this example, we asked the program to simulate 500 4-dice rolls. The sum was never four, the sum was five on 1 of the 500 trials, the sum was six on 3 of the trials, etc.

## 4 Procedure

You'll write this week's lab from scratch. Call your file `diceplot.cpp`.

### 4.1 [2 pts.] `int roll()`

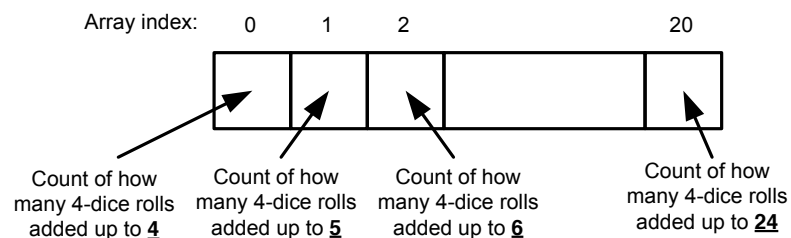
Define a function `roll()` that takes no arguments, and returns a random integer from 1 to 6. Before proceeding further, we recommend you write a short main method to test it out and make sure it gives you the right values. Comment this main out after you are satisfied that `roll()` works.

Remember: textbook Section 4.9 is on randomness, and Chapter 5 is on functions.

```
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
             // guaranteed to be random.
}
```

[HTTP://XKCD.COM/221/](http://xkcd.com/221/) (NOTE: THIS FUNCTION IS A JOKE...DON'T USE IT)

**Overview of the rest of the lab:** The rest of your lab will be to write a program to simulate 500 rolls of 4-dice.



### 4.2 [3 pts.] `void printHistogram(int counts[])`

Before moving on to the logic to add up rolls and count them, we will implement a function that encapsulates the output display of our sums. You will write a function `void printHistogram(int counts[])` that takes an array of integers (assumed to be of

size 21 elements), then prints out sequential labelled lines from **4 to 24** with as many `X` symbols on that line as is indicated in the appropriate element of the array (i.e. `counts[i]`). We start our labels at 4 since rolling 4 dice can't produce a sum of 0-3.

Let's test our `printHistogram` function by writing a sample `main()` that calls it. We will fill in a dummy array and just ask `printHistogram` to print it out. You should try the same and ensure your `printHistogram` is working with our dummy `main()`:

```
int main() {
    int testCounts[21]; // 21 options of sum of 4 dice => [4,24]
    for (int i=0; i<21; i++)
        testCounts[i] = i/2; // fill the array
    printHistogram(testCounts); // call your method
}
```

it should produce the following output:

```
4:
5:
6:X
7:X
8:XX
...
24:XXXXXXXXXX
```

### 4.3 [4 pts.] main function

Finally, write the main function of the program. (Comment out all the other test main methods you used earlier.) It should

- Ask the user for a number, call it **n**
- Run **n** experiments, where each experiment rolls 4 dice and adds them up
- Throughout the experiments, keeps track of how often each possible sum (from 4 to 24) occurred
- Prints out the histogram showing how many times each sum occurred (so in total it should print out **n** of the X symbols)
- Your program should give different results each time it runs. Make sure you seed the random number generator appropriately.

### 4.4 [1 pt.] Reflection

Suppose we wanted to change the program to roll 10 three-sided dice instead. Is there a different way to write your program to facilitate this change easily? (Hint: refer to page 39 of the textbook.)



*You can just answer this question to the TA verbally. You don't need to change your program.*

**Demonstrate your program and show your TA/CP the two functions you wrote explaining how it works.**