CMSC113 – COMPUTER SCIENCE 1
Assignment 3 – Due on Wednesday, March 5, 2025

In this assignment, you will write **two Java programs** that use arrays. Please be sure to review your Lab 4, class notes, and Section 1.4 from your text before attempting to work on this assignment. We strongly recommend that you design and implement the program in steps, incrementally. Start with the smallest program that you can write that does something small, but meaningful. Then, add to it to complete the given problem, incrementally. In both programs below, we provide some guidance as to how this will be done.

1. Birthday Problem
Suppose that people enter an empty room until a pair of people share a birthday. Assume birthdays to be uniform random integers between 0 and 364. On average, how many people would have to enter before there is a match?

To answer this question, first write a Java program that answers the question for one trial: how many people would have to enter before there is a match.

```
$ java-introcs BirthdayMatch 365
31

$ java-introcs BirthdayMatch 365
22
```

`BirthdayMatch` takes one command line argument that denotes the number of birthdays (it'll be 0 to 364, for 365 where 0 represents January 1 and 364 is December 364). You will need an array of 365 elements to record if you have already encountered that birthday or not (use true/false). You can then use this array to detect if you have already encountered a given birthday. Assume that birthdays are uniformly distributed from [0..364]. Here is an algorithm to do this part:

```
1. Input n (number of days in a year, i.e. 365)
2. Define a boolean array called, birthday of size, n.
3. Initialize all its elements to false. I.e. no matches yet.
4. Set peopleCount ← 0 [How many people have entered the room]
5. Repeat until a match is found
5.1 Set day ← a random number in [0..n) – a random birthday
5.2  Set peopleCount ← peopleCount + 1
     [Since a new person entered the room with birthday, day]
5.3  if  birthday[day] = true then
5.3.1   a match is found [we've seen this day before]
5.3  else
5.4.1   Set birthday[day] ← true [record the birthday]
6. Output peopleCount
```

Once done, modify your program to do a repeated number of trials and compute average.

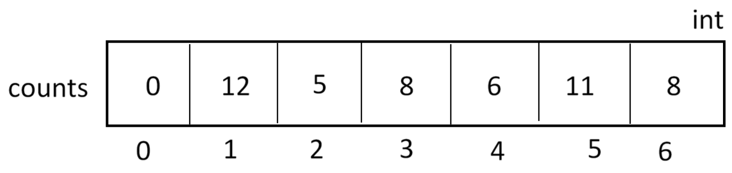
```
$ java-introcs BirthdayMatch 365 100000
22.34
```

Once done, **show the output for 10,000,000 trials.**

2. Dice roll histogram

Write a program that simulates the rolling of a six-sided die, **n** times. Once done, it prints out a histogram as shown below:

```
$ java-introcs Histogram 50
1: 12   XXXXXXXXXXXX
2: 5    XXXXX
3: 8    XXXXXXXX
4: 6    XXXXXX
5: 11   XXXXXXXXXXX
6: 8    XXXXXXXX
```



The program inputs the number of rolls (50) and then prints out the histogram. For example, in the above run there were 6 rolls of 4, 8 rolls of 6, etc. As in the **`BirthdayMatch`** program, implement this one incrementally. First, write a program that simulates all the rolls and prints out the counts (only) (see column 2 in output above). As shown in the illustration, you can create an array called **`counts`** where **`counts[i]`** contains the number of times the number **`i`** was rolled. Next, modify it so it prints out the histogram (the X's) using the **`counts`** array.

Show the output of your program for three runs of 50 rolls.

Producing outputs for submission

Once done, show sample runs of the two programs as specified above. You can feel free to cut and paste the program's output into a text document and then print it.

What to hand in:

- A printout of the two program files (the final version only).
- A printout showing sample runs of your program: **use sample inputs provided above, and the ones requested for each program**.
- Write and print out a short 1/2-paragraph reflection on your thoughts (how this assignment went for you, what went wrong, well, etc.).