Lab 11 - Playing with Arrays (dice)
CS 161 - Fall 2016

Due date: 4pm, Friday, December 9

Goals
- to declare and use arrays to remember information that must be handled more than once
- to use a random number generator
- to write several methods and loops to manipulate arrays

Overview
For this lab, you will write a program that (conceptually) rolls five dice and then evaluates the results for number of each possible value seen, total value, and poker/Yahtzee combinations (5 of a kind, 4 of a kind, full house, and so on). A seeded random number generator is used to provide repeatable values. Evaluating straights is an extra credit option.

Specification
Your program will record the results of rolling 5 dice. We will have a dice array of 5 integer values that is filled by calling a random integer generator five times. Each time, it should generate a number between 1 and 6. Your program should then print the following:

This program should use the Random class. You should pass the constructor a long integer to provide the initial seed. For example, if you have a Scanner, input, you could initialize a Random object called rand by:

```java
long seed = input.nextLong();
Random rand = new Random(seed);
```

You can use the nextInt(n) method on the Random object to obtain an integer in the range 0 to n-1. Adding one to that value will give you a value from 1 to n. For example, to get the value of a roll of the dice you could do something like:

```java
int roll = rand.nextInt(NUM_SIDES) + 1;
```

Your program should provide the following information (see samples below):
- the seed value used
- the 5 values rolled
- the number of times each of the possible dice values was rolled
- the total value of the dice rolled; ex., two 3s, two 4s, and a 5 would total 19
- a report of the best multiple value combination (5 of a kind, 4 of a kind, full house (3 of a kind of one value and 2 of a kind of another), 3 of a kind, two pair (two different 2 of a kind), 2 of a kind, or nothing) along with which dice value gave that combination; ex., 3 of a kind with 6s
- (10 pts extra credit) a report of a long straight (5 values in a row) or a short straight (4 values in a row)

Input Specification
The input will be a single, positive long integer used as a seed to start the random number generator at a known place so that we get repeatable results. For example, if the seed is 9142, the first five six-sided dice values are 4 5 6 2 2.

Output Specification
The output should contain the information listed in the specification section. It should be printed neatly and labelled. Your output should match the format shown in the example.

Data Representation
One aspect of problem solving is choosing an appropriate data representation. This choice may make your solution easier or harder to build. In this problem, it is helpful to have two arrays that have their useful information in elements 1 through 5 or 6 while ignoring the element with index 0. If you start the indexing at zero, your program will need to make several adjustments that make the program harder to understand. Wasting two integer memory locations is an excellent tradeoff. Therefore, consider the picture below showing the rolled and frequency arrays.

```
rolled:  5 2 5 2 2
        0 1 2 3 4 5 6
        indexes
frequency:  0 0 3 0 0 2 0
           0 1 2 3 4 5 6
           indexes
```

The first array, rolled, holds the values rolled. This is filled by generating five random values and placing each value in a separate element of the array. The second array, frequency, represents how many times each of the possible dice values were rolled. Note that the values in frequency were set to zero initially. rolled[0] has an meaningless value.

Declaring constants, i.e. class level public static final int, for NUM_SIDES (6, the number of sides on the dice) and NUM_DICE (5, the number of dice rolled) is a good idea. You may use these constant values in your methods. After you declare the rolled array to have a size of NUM_DICE+1 and the frequency array of size NUM_SIDES+1, you can run loops from 1 to < length.
Approaching the Problem
You will be overwhelmed if you do not develop your solution incrementally and use several methods.
We suggest the following iterations:

- Obtain the seed from the user, print it for reference, and then create the Random number generator using the seed.
  Generate several random values. Test to see that seed values give you repeatable generated values but also different generated values with different seeds.
- Declare an array to hold the results of rolling the dice five times (size NUM_DICE+1 to ignore the first element), fill it, and report which roll had which dice value (a general printArray method would be a good idea).
- Declare an array, frequency or freq, to show how many times each dice value was rolled (size NUM_SIDES+1), fill it based on the values in the other array, and print the results (here's where you re-use printArray). In filling this array, look at the rolled dice value and increment the corresponding freq element. This should be one line inside your loop.
- Total the dice and print the result
- Find the multiples (hints below)
- Find the straights (hints below; remember it is extra credit)

Hints
To find the multiples, use the frequency array. A five of a kind will have a 5 in one of that array’s elements. A four of a kind will have a 4. A method to evaluate whether x of a kind is found would be useful. Solve the full house and two pair portions after you have the plain kind working.

A method called ofAKind with the following description and header is useful. It should use a linear search that starts at the end of the array and runs backwards. If it returns zero, num was not found in A. But if it has a non-zero return value, that is where num of a kind was found. Use a starting point other than NUM_SIDES to search for a two of a kind after finding a three of a kind or another pair.

```java
public static int ofAKind(int num, int[] A, int start)
```

// returns where num of a kind was found in array A
// returns (largest <= start) index where found; returns 0 if not found
To find the straights, use the frequency array. A long straight will have a one (positive value) in five consecutive elements; for example, elements 1, 2, 3, 4, 5 or 2, 3, 4, 5, 6. A short straight has four consecutive positive elements; it may have a 2 in one of those elements. A method to find a positive run of a specified length starting at a given value would be helpful.

Delivery
When you are finished, submit your program as the dice assignment.

Sample Output (seed is the only input)
Enter seed: 1
The rolled dice: 1-> 4 2-> 5 3-> 2 4-> 4 5-> 3
The frequencies: 1-> 0 2-> 1 3-> 1 4-> 2 5-> 1 6-> 0
The dice totaled 18
Two of a kind with 4s
Short straight: 2 3 4 5

or

Enter seed: 32
The rolled dice: 1-> 6 2-> 2 3-> 6 4-> 2 5-> 2
The frequencies: 1-> 0 2-> 3 3-> 0 4-> 0 5-> 0 6-> 2
The dice totaled 18
Full house with three 2s and two 6s

Seed Hints (to get a set of five dice rolls with these characteristics, use the seeds shown)
5 of a kind: 407, 651, 2907, 3805 (these are hard to come by)
4 of a kind: 45, 55, 95, 101, 161, 214, 229, 279, 284, 372, 455, 462, 547
Full house: 32, 52, 124, 153, 155, 159, 162, 231, 253, 273, 281, 314, 329
3 of a kind: 7, 11, 14, 18, 28, 84
Two pair: 3, 4, 6, 28, 25, 27, 42, 47, 53
2 of a kind: 5, 12, 24, 40, 43, 44, 58
Long straight: 2, 68, 108, 172
Short straight: 37, 85, 129, 222
Short straight and 2 of a kind: 1, 15, 23, 35, 48, 139, 234
Nothing: 104, 137, 142, 166, 219, 266