APCS Programming Lab 7 – Arrays

80% level

1. Write and test a class called ArrayClass which will have the following methods:
   a. The constructor public **ArrayClass(int n)** where n is the length of the array as provided by the user in the test program
   b. **public void fillArray(int n)** which will fill the array with random integers between 0 and n (inclusive), where n is specified by the user in the test program
   c. **public int findMax()** which will find and return the maximum value in the array
   d. **public int findMin()** which will find and return the minimum value in the array
   e. **public int sumArray()** which will find and return the sum of the integers in the array
   f. **public double avgArray()** which will find and return (as a double) the average of the integers in the array
   g. **public void displayArray()** which will print out the list of integers in the array
   h. **public int valueAt(int n)** which will return the integer stored at index n in the array (supplied by user in the test program)
   i. **public int findValue(int n)** which will search the array for the integer n (supplied by user in the test program) and return the index of the first location where n is found or return an appropriate message is n is not in the array. This is referred to as a "linear search".
   j. **public void reverseArray()** which will reverse the numbers in the array.

2. Construct a class DiceRoll with the following specifications:
   a. The constructor **DiceRoll(int n)** will take one argument, which is the number of times a pair of dice will be rolled. (The user will be prompted in the test program to enter this number.)
   b. The public void method **roll()** will roll the pair of dice the specified number of times and use an array to keep track of how many sums of 2, 3…12 are rolled.
   c. The public int method **getNumber(int n)** will return how many times the specified values was rolled.
   d. The public void method **displayResults()** will print the results of the rolls:
      2's: 5
      3's: 7
      …
      12's: 6
   e. The public void method **displayHistogram()** will display the results in the form of a histogram. You may choose to:
      i. print out stars (horizontally)

ii. use graphics to make a histogram
iii. (bonus) print out stars vertically

3. Write a class, FishSwim, which will simulate a fish swimming in a one-dimensional tank.. The tank will be represented by an array of user-defined length, filled with one-character strings: "_" for no fish and "*" for a fish. The fish will start at a random position. For each iteration of the simulation, the fish will either swim one unit to the left or one unit to the right. If the fish is at one end of the tank it will always swim one unit away from the end. The user will supply the number of iterations for the simulation to be run as well as the probability of swimming to the right on each iteration.

   public FishSwim(int len, double p) is the constructor where len is the length of the tank and p is the probability of swimming to the right. The initial starting position of the fish is also to be determined in the constructor.

   private void swimOneUnit() is a method which decides which way the fish swims, taking into account the probability of swimming to the right and the ends of the tank and updates the array with the new location of the fish

   private void displayTank() displays the present state of the tank

   public void swim(n) runs the simulation n times

   The test program will prompt the user for the length of the tank, the probability of swimming to the right and number of times the simulation will be run. Then the tank will be constructed and the swim(int n) method called once. The output will look something like this:

   ---------*-
   ----------*
   ---------*-
   --------*--
   -------*---
   --------*--
   …

4. Implement the method **public void sortArray()** to sort the array from problem 1 from smallest to largest. You may not use any built in java methods.

5. Write and test a class called TwoD which will have the following methods:

a. The constructor public **TwoD(int n, int m)** where n is the number of rows in a two-dimensional array and m is the number of columns in a two-dimensional array as provided by the user in the test program

b. **public void fillTwoDArray(int n)** which will fill the array with random integers between 0 and n (inclusive), where n is specified by the user in the test program
c. **public void displayArray()** which will print out the elements in the array in matrix format

90% level

6. **The Locker Problem:** You are in a locker room with 100 open lockers, numbered 1 to 100. Toggle all of the lockers that are even. By *toggle*, we mean close if it is open, and open if it is closed. Now toggle all of the lockers that are multiples of three. Repeat with multiples of 4, 5, up to 100. Display which lockers are open.
7. Magic Squares: An n-by-n matrix that is filled with the numbers 1,2,3,…,n^2 is a magic square if the sum of the elements in each row, in each column and in the two diagonals is the same value. An example of a magic square:

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Write a program that will accept values from a user and check for the following:
   a. Did the user enter n^2 values?
   b. Does each of the values 1,2,3,… n^2 occur once?
   c. If the values are entered into a n-by-n square matrix (row by row) do they form a magic square?

100% level

8. Construct and test a class Matrix which will have the following methods
a. The constructor **Matrix()**
b. Method void **fillMatrix()** which will prompt the user for dimensions and then elements.
c. Method void **transposeMatrix()** which swaps the rows and columns
d. Method void **addMatrix(Matrix a)** which adds this matrix to Matrix a, if the dimensions allow
e. Method void **subtractMatrix(Matrix a)** which subtracts this matrix minus Matrix a, if the dimensions allow
f. Method void **multiplyMatrix(Matrix a)** which multiplies this matrix with Matrix a, if the dimensions allow
g. Method void **displayMatrix()** which displays the matrix