CSCI 145 – Preparing for Lab Final

Make sure to read all instructions before attempting this assignment. You cannot work with another student or communicate with anyone for this assignment. If you submitted an online solution, a solution on site such as Discord, or someone else solution, 0 will be given. If you posted/shared a solution and someone else uses it, you might get a 0 as well. You will only need to submit 2 out of 3 problems.

Each problem is worth 20 points and it is best to work on all 3 problems first. You will need to modify and submit two problems based on the new requirements on lab final day.

1. One interesting application of two-dimensional arrays is special squares. **A special square is a square matrix in which values in each row and in each column are in non-descending order** (smallest to largest in all rows and columns but can be like 5 5 7 8). In this exercise you will write code to determine whether a square is a special square. Your program reads input for squares from a file named *specialData.txt* (need to create the data file) and tells whether each is a special square by printing yes or no. Note that the -1 at the bottom tells the test program to stop reading. You can write your program from scratch, but it is recommended that you use "Magic Squares" exercise from one PA as a guide.

   Sample input data for specialData.txt:

```
3
0 1 3
1 2 3
2 5 8
4
0 2 4 6
1 3 6 7
2 4 5 8
5 7 8 9
3
0 1 2
1 2 3
2 5 4
-1
```

   Sample output data for specialData.txt:

```
1 yes
2 no
3 no
```

2. Given the following recursive definition, implement a recursive method. You can assume that n and x are integers as well as n >= 0 and x != 0.

```
p(x, n) = 1 when n is 0
p(x, n) = p(x, n/2) ^ 2 when n is even
p(x, n) = x * p(x, n-1) when n is odd
```

   Try this recursive method with the following two test cases:

```
result = p(2, 9);
// use System.out.println to output result
// result: 512

result = p(2, 4);
// use System.out.println to output result
// result: 16
```

3. Define a Java class called **Changer** with the following methods (set up proper attributes and imagine this is for a money changer for quarters, dimes, and nickels). Pennies are not used, and we only deal with multiples of 5 cents.
   - a default constructor: takes no parameter; sets id to "Changer 1" and sets quarters, dimes, and nickels to 4.
   - an overload constructor: takes 4 parameters for id, quarters, dimes, and nickels; assume valid data.
   - getId: returns the id of this changer.
   - setId: takes a new id and replaces the current id.
   - balance: no parameter; returns the total amount in the changer as dollars and cents like 5.25.
   - makeChanges: takes an int parameter representing an amount (assume multiple of 5 and between 5 and 95); returns true if there are available coins to perform the transaction (maximize largest denomination and subtract each denomination as applicable); returns false otherwise (e.g., need 2 quarters and there are not enough quarters regardless availability of dimes and nickels) and do not modify coins in the changer.
   - toString: overrides ***toString*** method and returns applicable information.

   Set up a simple test driver to test your class. Run the following test cases:

```
Changer myChanger = new Changer();
System.out.println(myChanger);   // Changer 1, q = 4, d = 4, n = 4
System.out.println("amount: " + myChanger.balance());
                                 // amount: 1.60

boolean valid = myChanger.makeChanges(60);   // valid = true
System.out.println("valid: " + valid);        // valid: true
System.out.println(myChanger);   // Changer 1, q = 2, d = 3, n = 4

myCharger.setId("C1");
valid = myChanger.makeChanges(90);            // valid = false
System.out.println("valid: " + valid);        // valid: false
System.out.println(myChanger);   // C1, q = 2, d = 3, n = 4

Changer changer2 = new Changer("C2", 2, 1, 1);
System.out.println(changer2);    // output: C2, q = 2, d = 1, n = 1
valid = changer2.makeChanges(45);             // valid = false
System.out.println("valid: " + valid);        // valid: false
System.out.println(changer2);    // output: C2, q = 2, d = 1, n = 1
```