Langara College

Department of Computing Science and Information Systems

CPSC 1150 – Program Design **Final Exam**

Summer 2020

Time: 2 hours

Total Marks: 80

Name: **ID:**

Instructions:

1) Create a new folder named **Final** and save all your files inside this folder
2) For multiple choice questions, please finish them on D2L BrightSpace.
3) For all the written questions, scan your solution and **save as a pdf file**.
4) For all the programming problems, save the programs as the names provided. Feel free to add comments where necessary, but full internal documentation is not required.
5) The main method for the first two programming question is given. For the third programming question, you need to write a complete program. The input text file for the third programming question is also given.
6) At the end of the test, zip the folder **Final** and upload it to BrightSpace.
7) By taking this test, you declare you will not cheat.
8) You may not collaborate with anyone.
9) You may not ask for help from anyone.
10) You may not use any library functions that are not covered in this course.
11) You may use the Internet, **but you cannot search for solutions**.

I. **Multiple Choice Questions (15 marks. One mark each)** See D2L BrightSpace Fina Exam Quiz.

II. Fix Logic Errors (15 marks)

1. (6 marks) The following program contains **two** logic errors. The expected output of the program is shown below. Analyze and correct the logic errors in the code. While correcting the errors, the structure of the code **should not** be changed. This includes the number of loops and the type of the array.

   The expected output:

```
****
***B
**CB
*DCB
EDCB
```

```
public class Debug1 {
        public static void main(String[] args){
                char[] arr = {'A',  'B', 'C', 'D', 'E'};
                String tmp;
                int i, j;
                for (i = 0; i < arr.length; i++){
                        tmp = "";
                        for (j = arr.length - 1; j > 0; j--)
                                tmp += '*';

                        for (j = arr[0]; j < arr[i]; j++)
                                tmp += (char)j;

                        System.out.println(tmp);
                }
        }
}
```

2. (9 marks) The following program contains **three** logic errors. Find and fix them.

   The program displays five random numbers between (and including) user-specified values.

```
import java.util.Scanner;

public class Debug2{
  public static void main(String[] args){
          int high, low, count = 0;
          final int NUM = 5;
          Scanner input = new Scanner(System.in);

          // Prompt user to enter high and low values
          System.out.println("This application displays " + NUM + " random numbers" +
                  "\nbetween (and including) the low and high values you enter");

          System.out.print("\nEnter low value: ");
          low = input.nextInt();

          System.out.print("Enter high value: ");
          high = input.nextInt();

          while(high >= low){
                  System.out.println("The number you entered for high, " +
                          high + ", is not higher than " + low);
                  System.out.print("Enter a number higher than " + low + ": ");
                          high = input.nextInt();
          }

          while(count < high){
                  // Generate a random number between (and including) low and high
                  double result = Math.random();
                  int number = (int)(result * (high - low + 1));

                  System.out.print(number + "  ");
                  ++count;
          }
          System.out.println();
  }
}
```

III. Programming (50 marks)

1. (15 marks) (**RandomVerificationCode.java**) Write a Java method called *generateVefiCode* that returns a string of verification code of length m (m is a parameter passed to the method). Each character in the code is randomly generated and it must be a digit (0-9), an uppercase letter (A-Z), or a lowercase letter (a-z). In addition, the code must contain at least one digit, one uppercase letter and one lowercase letter. You may assume that the length m is greater than or equal to three. The main method is given.

2. (13 marks) (**MonthlyPrecipitation.java**) The table below gives the monthly precipitation for Vancouver during a five-year period. Write a Java method called *averagePrecips* that calculates and displays **the average precipitation over the five years for each month**. Assume that the precipitation data (not the year and month names) has been stored in a 2D array and this array will be passed as a parameter to your method. The main method is given. Use loops wherever possible.

Monthly Precipitation (in mm) for Vancouver

	Jan	Feb	Mar	Apr	May	June	July	Aug	Sept	Oct	Nov	Dec
2011	132	116	105	75	62	46	36	38	64	115	167	161
2012	133	118	115	73	63	48	34	35	65	114	163	160
2013	134	117	106	77	65	44	37	33	63	113	165	159
2014	135	113	108	72	62	47	39	34	67	112	164	162
2015	131	116	107	79	68	43	33	39	68	112	162	160

```
int[][] precips = {{132, 116, 105, 75, 62, 46, 36, 38, 64, 115, 167, 161},
                   {133, 118, 115, 73, 63, 48, 34, 35, 65, 114, 163, 160},
                   {134, 117, 106, 77, 65, 44, 37, 33, 63, 113, 165, 159},
                   {135, 113, 108, 72, 62, 47, 39, 34, 67, 112, 164, 162},
                   {131, 116, 107, 79, 68, 43, 33, 39, 68, 112, 162, 160}};
```

Please keep one digit after the decimal point. Your output should be like the following:

```
Average precipitation for each month
1  133.0
2  116.0
3  108.2
4  75.2
5  64.0
6  45.6
7  35.8
```

8	*35.8*
9	*65.4*
10	*113.2*
11	*164.2*
12	*160.4*

3. (22 marks) **(selectMagicNumber.java**) Use top-down design methodology to develop a complete Java program to read integers (n) from *input.txt* file. The program should reject those integers (n) which does not match the following specifications:

- n should not contain the digit zero
- None of the digits of n should be repeated, the digits should be different
- 2*n should have the same digits as n (order of digits can be different) and should not have the digit zero in it as well as none of digits be repeated.

Few Examples:

- **425871 is a valid number**.
  (It does not contain digit zero
  none of the digits are repeated,
  2* 425871 = 851742 (2*n) has the same digits as n (1,2,4,5, 7, and 8))

- **428071 is not valid.**
  (It contains digit zero)

- **1215789 is not valid.**
  (Digit 1 is repeated)

- **124587 is not valid.**
  (Though all digits of the number are unique, and it has no zero digits, but 2*124587 = 2**49**17**4,** does not have the same digits as n. n has digit 8 while 2*n has digit 9, and digit 4 is repeated in 2*n)

Finally, save the result in a column format in *result.txt* file as shown below (The 1st Row is the Title, whereas the rest of the rows are the filtered output satisfying the specifications as above:

```
    n        2*n
--------------------
    n        2n
```

A sample input.txt file and result.txt file is shown below:

sample input.txt file: **note** that bold numbers are the only valid numbers listed in the file as per the specifications mentioned above.

124587 412507 412852 ***425871*** *428071 1205807 1207859 1244579* ***1247859*** *1248769 1258874* ***1259874*** *1259848 1275489 15867432 17246358 17256342* ***17264358*** *17564123 17635832 17823866* ***17824536*** *17834562 17824356 17865324 17878342 32577864 32685571 32741186 32741850* ***32856417*** *32867554 38256147*

result.txt file

```
       n         2n
--------------------
    425871    851742
   1247859   2495718
   1259874   2519748
  17264358  34528716
  17824356  35648712
  32856417  65712834
```

Notes:

1. You should split your program into simple methods, and each method should do a simple job.
2. You will be marked based on correctness and quality of your code.
3. Your program will be tested with a different input.txt file than the sample shown above. Hence, come up with a generic code that works for any input.txt file containing random integer numbers.
4. Do not worry too much about the exact format when displaying numbers in result.txt
5. **Hint: You may want to use an array to store the number of occurrences of each digit in n.**