FindStringCode

Crazy Zak has designed the below steps which can be applied on any given string (sentence) to produce a number.
STEP1. In each word, find the Sum of the Difference between the first letter and the last letter, second letter and the penultimate letter, and so on till the center of the word.
STEP2. Concatenate the sums of each word to form the result.
For example –
If the given string is “WORLD WIDE WEB”
STEP1. In each word, find the Sum of the Difference between the first letter and the last letter, second letter and the penultimate letter, and so on till the center of the word.
WORLD = [W-D]+[O-L]+[R] = [23-4]+[15-12]+[18] = [19]+[3]+[18] = [40]
WIDE = [W-E]+[I-D] = [23-5]+[9-4] = [18]+[5] = [23]
WEB = [W-B]+[E] = [23-2]+[5] = [21]+[5] = [26]
STEP2. Concatenate the sums of each word to form the result
[40] [23] [26]
[402326]
The answer (output) should be the number 402326.
NOTE1:The value of each letter is its position in the English alphabet system i.e. a=A=1, b=B=2, c=C=3, and so on till z=Z=26.
So, the result will be the same for “WORLD WIDE WEB” or “World Wide Web” or “world wide web” or any other combination of uppercase and lowercase letters.
IMPORTANT Note: In Step1, after subtracting the alphabets, we should use the absolute values for calculating the sum. For instance, in the below example, both [H-O] and [E-L] result in negative number -7, but the positive number 7 (absolute value of -7) is used for calculating the sum of the differences.
Hello = [H-O]+[E-L]+[L] = [8-15]+[5-12]+[12] = [7]+[7]+[12] = [26]
Assumptions: The given string (sentence) will contain only alphabet characters and there will be only one space character between any two consecutive words.
You are expected to help Zak, by writing a function that takes a string (sentence) as input, performs the above mentioned processing on the sentence and returns the result (number).
Example1:
input1 = “World Wide Web”
output1 = 402326
Example2:
input1 = “Hello World”
output1 = 2640
Explanation:
Hello = [H-O]+[E-L]+[L] = [8-15]+[5-12]+[12] = [7]+[7]+[12] = [26]
World = [W-D]+[O-L]+[R] = [23-4]+[15-12]+[18] = [19]+[3]+[18] = [40]
Result = Number formed by concatenating [26] and [40] = 2640

```java
import java.util.Scanner;

public class FindStringCode{
    public static int findStringCode(String input1)
    {
        String upperCase=input1.toUpperCase();
        String str[]=upperCase.split(" ");
        String output="";
        for(int i=0;i<str.length;i++)
        {
            int sum=0;
            for(int j=0;j<str[i].length()/2;j++)
            {
                int first=str[i].charAt(j)-64;
                // System.out.println(first);
                int last=str[i].charAt(str[i].length()-1-j)-64;
                // System.out.println(last);
                sum+=Math.abs(first-last);
                // System.out.println(sum);
            }
            if(str[i].length()%2!=0)
            {
                sum+=str[i].charAt(str[i].length()/2)-64;
            }
            output=output+sum;
        }
        return Integer.parseInt(output);
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(findStringCode(s));
    }
}
```

Get Code Through Strings:

Farah is one of the few associates in Global Safe Lockers Corp Limited, who has access to the company's exclusive locker that holds confidential information related to her division. The PIN to the locker gets changed every two days. Farah receives the PIN in the form of a string which she needs to decode to get the single-digit numeric PIN.
The numeric PIN can be obtained by adding the lengths of each word of the string to get the total length, and then continuously adding the digits of the total length till we get a single digit.
For example, if the string is "Wipro Technologies", the numeric PIN will be 8.
Explanation:
Length of the word "Wipro" = 5
Length of the word "Technologies" = 12
Let us add all the lengths to get the Total Length = 5 + 12 = 17
The Total Length = 17 , which is not a single-digit, so now let us continuously add all digits till we get a single digit i.e. 1+ 7 = 8
Therefore, the single-digit numeric PIN = 8
Farah approaches you to write a program that would generate the single-digit numeric PIN if the string is input into the program. Help Farah by writing the function (method) that takes as input a string input1 that represents the sentence, and returns the single-digit numeric PIN.
Assumptions: For this assignment, let us assume that the given string will always contain more than one word.
Let's see one more example -
If the given string is "The Good The Bad and The Ugly", the numeric PIN would be = 5
Explanation:
Let us add lengths of all words to get the Total Length = 3+4+3+3+3+3+4 = 23
Total Length = 23, which is not yet a single digit, so let us continue adding all digits of the Total Length, i.e. 2+3 = 5
Therefore, single-digit numeric PIN = 5

```java
import java.util.Scanner;
public class GetThroughString {
    public static int SingleDigitSum(int digit)
    {
        int sum=0;
        while(digit>0||sum>9)
        {
            if(digit==0)
            {
                digit=sum;
                sum=0;
            }
            sum+=digit%10;
            digit/=10;
        }
        return sum;
    }
    public static int getCodeThroughString(String input1)
    {
        int sum=0;
        String str[]=input1.split(" ");
        for(int i=0;i<str.length;i++)
        {
            sum+=str[i].length();
        }
        // return (1+(sum-1)%9);
        return SingleDigitSum(sum);
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String str=sc.nextLine();
        System.out.println(getCodeThroughString(str));
    }
}
```

Addition using Strings: Write a function that takes two numbers in string format and forms a string containing the sum (addition) of these two numbers.

Assumption(s):
The input strings will contain only numeric digits
The input strings can be of any large lengths
The lengths of the two input string need not be the same
The input strings will represent only positive numbers
For example –
If input strings are “1234” and “56”, the output string should be “1290”
If input strings are “56” and “1234”, the output string should be “1290”
If input strings are “123456732128989543219” and “987612673489652”, the output string should be “123457719741663032871”
NOTE: In Java & C#, this logic can be easily implemented using BigInteger. However for the sake of enhancing your programming skills, you are recommended to solve this question without using BigInteger.

```java
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Scanner;

public class AdditionUsingString {
    public static String addNumberString(String s1,String s2)
    {
        BigDecimal b1=new BigDecimal(s1);
        BigDecimal b2=new BigDecimal(s2);
        return String.valueOf(b1.add(b2));
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1,s2;
        s1=sc.nextLine();
        s2=sc.nextLine();
        System.out.println(addNumberString(s1, s2));
    }
}
```

Simple Encoded Array:

Maya has stored few confidential numbers in an array (array of int). To ensure that others do not find the numbers easily, she has applied a simple encoding.
Encoding used : Each array element has been substituted with a value that is the sum of its original value and its succeeding element's value.
i.e. arr[i] = original value of arr[i] + original value of arr[i+1]
e.g. value in arr[0] = original value of arr[0] + original value of arr[1]
Also note that value of last element i.e. arr[last index] remains unchanged.
For example,
If the encoded array is {7,6,8,16,12,3}
The original array should have been {2,5,1,7,9,3}
Provided the encoded array, you are expected to find the –
First number (value in index 0) in the original array
Sum of all numbers in the original array
Write the logic in the function findOriginalFirstAndSum(int[] input1, int input2);
where,
input1 represents the encoded array, and
input2 represents the number of elements in the array input1
The method is expected to –
find the value of the first number of the original array and store it in the member output1 and
find the sum of all numbers in the original array and store it in the member output2
Note that the output1 and output2 should be returned as -
- members of a Result object (if the code is being written in Java, C# or C++)
- members of a Result struct (if the code is being written in C)
Assumption: The array elements can be positive and/or negative numbers
Example 1:
If the encoded array is {7,6,8,16,12,3}
The Original array should have been {2,5,1,7,9,3}
So, First number in original array = 2
Sum of all numbers in original array = 27
Example 2:
If the encoded array is {-2,-7,-12,-15}
The Original array should have been {8,-10,3,-15}
So, First number in original array = 8
Sum of all numbers in original array = -1

```java
import java.util.Scanner;

class SimpleEncodedArray {
    public class Result
    {
        public final int OUTPUT1;
        public final int OUTPUT2;
        public Result(int out1,int out2)
        {
            OUTPUT1=out1;
            OUTPUT2=out2;
        }
    }
    public Result findOriginalFirstAndSum(int input1[],int
input2){
        int array[]=new int[input2];//6
        array[input2-1]=input1[input2-1];
        int sum=array[input2-1];
        for(int i=input2-2;i>=0;i--){
            array[i]=input1[i]-array[i+1];
            sum+=array[i];
        }
        Result r=new Result(array[0], sum);
        return r;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int array[]={7,6,8,16,12,3};//6
        SimpleEncodedArray s=new SimpleEncodedArray();
        Result r=s.findOriginalFirstAndSum(array,
array.length);
        System.out.println(r.OUTPUT1);
        System.out.println(r.OUTPUT2);
    }
}
```

Decreasing sequence:

Given an integer array, find the number of decreasing sequences in the array and the length of its longest decreasing sequence.
You are expected to complete the logic within the given function,
where,
input1 represents the integer array and,
input2 represents the number of elements in the integer array
The function should set the output1 variable to the number of decreasing sequences in the array, and set the output2 variable to the length of the longest decreasing sequence in the array.
Example 1:
If input1[ ] = {11,3,1,4,7,8,12,2,3,7}
and input2 = 10
output1 should be 2
output2 should be 3
Explanation:
In the given array input1, the decreasing sequences are “11,3,1” and “12,2”, i.e. there are two decreasing sequences in the array, and so output1 is assigned 2. The first sequence i.e. “11,3,1” is the longer one containing three items, when compared to the second sequence “12,2” which contains 2 items. So, the length of the longest decreasing sequence output2 = 3.
Example 2:
If input1[ ] = {9}
and input2 = 1
output1 should be 0
output2 should be 0
Example 3:
If input1[ ] = {12,51,100,212,15,12,7,3,57,300,312,78,19,100,102,101,99,74,0,-5}
and input2 = 20
output1 should be 3
output2 should be 6

```
import java.util.Scanner;

class DecreasingSequence {
    public class Result {
        public final int OUTPUT1;
        public final int OUTPUT2;
        public Result(int out1, int out2) {
            OUTPUT1 = out1;
```

```
            OUTPUT2 = out2;
        }
    }
    public Result decreasingSeq(int input1[], int input2) {
        int decreaseCount = 0;
        int LongCount = 0;
        int SpikeCount = 0;
        boolean flag = false;
        for (int i = 0; i < input2 - 1; i++) {
            if (input1[i] > input1[i + 1]) {
                if (flag == false) {
                    flag = true;
                    SpikeCount++;
                }
                decreaseCount++;
                LongCount = decreaseCount > LongCount ?
decreaseCount : LongCount;
            }
            else
            {
                if(flag==true)
                {
                    flag=false;
                    decreaseCount=0;
                }
            }
        }
        if(SpikeCount>0)
            LongCount++;
        return new Result(SpikeCount, LongCount);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int array[] = { 11, 3, 1, 4, 7, 8, 12, 2, 3, 7 };
        Result r=new DecreasingSequence().decreasingSeq(array,
array.length);
```

```
        System.out.println(r.OUTPUT1);
        System.out.println(r.OUTPUT2);
    }
}
```

Find the Most Frequently Occurring Digit in a series of numbers.

Kamal is a data analyst in a lottery management organization.
One of the tasks assigned to Kamal is to find the Most Frequently Occurring Digit in a series of input numbers.
Below are a couple of examples to illustrate how to find the Most Frequently Occurring Digit in a series of input numbers.
Example1 –
If the series of input numbers are [1237, 262, 666, 140]
We notice that,
0 occurs 1 time
1 occurs 2 times
2 occurs 3 times
3 occurs 1 time
4 occurs 1 time
5 occurs 0 times
6 occurs 4 times
7 occurs 1 time
8 occurs 0 times
9 occurs 0 times
We observe that –
4 is the highest frequency in this series, and,
6 is the digit that occurs 4 times.
Thus, the Most Frequently Occurring Digit in this series is 6.
NOTE: If more than one digit occur the most frequent time, then the largest of the digits should be chosen as the answer. See below example for clarity on this point.
Example2 –
If the series of input numbers is [1237, 202, 666, 140]
We notice that,
0 occurs 2 times
1 occurs 2 times
2 occurs 3 times

3 occurs 1 time
4 occurs 1 time
5 occurs 0 times
6 occurs 3 times
7 occurs 1 time
8 occurs 0 times
9 occurs 0 times
We observe that –
3 is the highest frequency in this series, and,
2 and 6 are the digits that occur 3 times.
The larger of the two digits (2 and 6) is 6. Thus, the Most Frequently Occurring Digit in this series is 6.
Help Kamal by writing the logic in the function mostFrequentlyOccurringDigit for finding the Most Frequently Occurring Digit in the provided series of numbers.
The function takes two inputs -
input1 is the array of numbers
input2 is the number of elements in the array input1

```
public class MostFrequencyOccuringDigitInASeriesOfNumber {
    public static int
mostFrequencyOccuringDigitInASeriesOfNumber(int input1[],int
input2)
    {
        int max=Integer.MIN_VALUE;
        int highestDigit=Integer.MIN_VALUE;
        int array[]=new int[10];
        for(int i=0;i<input2;i++)
        {
            while (input1[i]!=0) {
                array[input1[i]%10]++;
                input1[i]/=10;
            }
        }
        for(int i=0;i<array.length;i++)
        {
            if(array[i]>=max)
            {
```

```
                max=array[i];
                highestDigit=i;
            }
        }
        return highestDigit;
    }
    public static void main(String[] args) {
        int array[]={1237, 202, 666, 140};
System.out.println(mostFrequencyOccuringDigitInASeriesOfNumber(
array, array.length));
    }
}
```

Sum of Powers of Digits:

Alex has been asked by his teacher to do an assignment on powers of numbers. The assignment requires Alex to find the sum of powers of each digit of a given number, as per the method mentioned below.
If the given number is 582109, the Sum of Powers of Digits will be calculated as =
= (5 raised to the power of 8) + (8 raised to the power of 2) + (2 raised to the power of 1) + (1 raised to the power of 0) + (0 raised to the power of 9) + (9 raised to the power of 0)
i.e. each digit of the number is raised to the power of the next digit on its right-side. Note that the right-most digit has to be raised to the power of 0. The sum of all of these powers is the expected result to be calculated.
Example - If the given number is 582109, the Sum of Powers of Digits =
= (5 raised to the power of 8) + (8 raised to the power of 2) + (2 raised to the power of 1) + (1 raised to the power of 0) + (0 raised to the power of 9) + (9 raised to the power of 0)
= 390625 + 64 + 2 + 1 + 0 + 1 = 390693
Alex contacts you to help him write a program for finding the Sum of Powers of Digits for any given number, using the above method.
Write the logic in the given function sumOfPowerOfDigits where,
input1 represents the given number.
The function is expected to return the "Sum of Powers of Digits" of input1.
Assumptions: For this assignment, let us assume that the given number will always contain more than 1 digit, i.e. the given number will always be >9.

```java
import java.util.Scanner;
public class SumOfPowerOFDigits {
    public static int sumOfPowerOFDigits(int input1)
    {
        char ch[]=String.valueOf(input1).toCharArray();

        int sum=0;
        for(int i=0;i<ch.length-1;i++)
        {
            int x=Integer.parseInt(String.valueOf(ch[i]));
            int y=Integer.parseInt(String.valueOf(ch[i+1]));
            sum+=Math.pow(x, y);
        }
        return
sum+(int)Math.pow(Integer.parseInt(String.valueOf(ch[ch.length-
1])), 0);
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        System.out.println(sumOfPowerOFDigits(n));
    }
}
```

Sum of Sums of Digits in Cyclic order:

Alex has been asked by his teacher to do an assignment on sums of digits of a number. The assignment requires Alex to find the sum of sums of digits of a given number, as per the method mentioned below.
If the given number is 582109, the Sum of Sums of Digits will be calculated as =
= (5 + 8 + 2 + 1 + 0 + 9) + (8 + 2 + 1 + 0 + 9) + (2 + 1 + 0 + 9) + (1 + 0 + 9) + (0 + 9) + (9)
= 25 + 20 + 12 + 10 + 9 + 9 = 85
Alex contacts you to help him write a program for finding the Sum of Sums of Digits for any given number, using the above method.

Help Alex by completing the login in the given function sumOfSumsOfDigits which takes as input an integer input1 representing the given number.
The function is expected to return the "Sum of Sums of Digits" of input1.
Assumptions: For this assignment, let us assume that the given number will always contain more than 1 digit, i.e. the given number will always be >9.

```
import java.util.Scanner;
public class SumOfSumsOfDigitsInCyclicOrder {
    public static int sumOfSumsOfDigitsInCyclicOrder(int
input1) {
        String str = Integer.toString(input1);
        int sum = 0;
        for (int i = 0; i < str.length(); i++) {
            for (int j = i; j < str.length(); j++) {
                int num =
Character.getNumericValue(str.charAt(j));
                sum += num;
            }
        }
        return sum;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(sumOfSumsOfDigitsInCyclicOrder(n));
    }
}
```

Identify possible words:

Detective Bakshi while solving a case stumbled upon a letter which had many words whose one character was missing i.e. one character in the word was replaced by an underscore. For e.g.“Fi_er”. He also found thin strips of paper which had a group of words separated by colons, for e.g. “Fever:filer:Filter:Fixer:fiber:fibre:tailor:offer”. He

could figure out that the word whose one character was missing was one of the possible words from the thin strips of paper. Detective Bakshi has approached you (a computer programmer) asking for help in identifying the possible words for each incomplete word.

You are expected to write a function to identify the set of possible words.
The function identifyPossibleWords takes two strings as input
where,

input1 contains the incomplete word, and
input2 is the string containing a set of words separated by colons.

The function is expected to find all the possible words from input2 that can replace the incomplete word input1, and return the result in the format suggested below.

Example1 -

input1 = “Fi_er”
input2 = “Fever:filer:Filter:Fixer:fiber:fibre:tailor:offer”

output string should be returned as “FILER:FIXER:FIBER”
Note that –

The output string should contain the set of all possible words that can replace the incomplete word in input1
all words in the output string should be stored in UPPER-CASE
all words in the output string should appear in the order in which they appeared in input2, i.e. in the above example we have FILER followed by FIXER followed by FIBER.
While searching for input1 in input2, the case of the letters are ignored, i.e “Fi_er” matches with “filer” as well as “Fixer” as well as “fiber”.
IMPORTANT: If none of the words in input2 are possible candidates to replace input1, the output string should contain the string “ERROR-009”

Assumption(s):

Input1 will contain only a single word with only 1 character replaced by an underscore “_”
Input2 will contain a series of words separated by colons and NO space character in between
Input2 will NOT contain any other special character other than underscore and alphabetic characters.

```
import java.io.*;
import  java.util.*;
class UserMainCode
{
    public String identifyPossibleWords(String input1,String input2){
        String output="";
        String array[]=input2.split(":");
        int index=input1.indexOf("_");
        for(int i=0;i<array.length;i++)
        {
            if(array[i].length()==input1.length())
            {
                String test1=array[i].toUpperCase();
                char ch[]=test1.toCharArray();
                ch[index]='_';
                test1=String.valueOf(ch);
                String test2=input1.toUpperCase();
                if(test2.equals(test1))
                {
                    if(output=="")
                    {
                        output+=array[i].toUpperCase();
                    }
                    else
                    {
                        output+=":"+array[i].toUpperCase();
                    }
                }
            }
        }
```

```
        if(output=="")
        {
            output="ERROR-009";
        }
        return output;
    }
}
public class IdentifyPossibleWords
{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        UserMainCode um=new UserMainCode();
        System.out.println(um.identifyPossibleWords("Fi_er",
"Fever:filer:Filter:Fixer:fiber:fibre:tailor:offer"));
    }
}
```

Encoding Three Strings:

Anand was assigned the task of coming up with an encoding mechanism for any given three strings. He has come up with the below plan.
STEP ONE: Given any three strings, break each string into 3 parts each.
For example – If the three strings are as below -
Input1= “John”
Input2= “Johny”
Input3= “Janardhan”
“John” should be split into “J”, “oh”, “n” as the FRONT, MIDDLE and END parts respectively.
“Johny” should be split into “Jo”, “h”, “ny” as the FRONT, MIDDLE and END parts respectively.
“Janardhan” should be split into “Jan”, “ard”, “han” as the FRONT, MIDDLE and END parts respectively.
i.e. if the no. of characters in the string are in multiples of 3, then each split-part will contain equal no. of characters, as seen in the example of “Janardhan”

If the no. of characters in the string are NOT in multiples of 3, and if there is one character more than multiple of 3, then the middle part will get the extra character, as seen in the example of “John”
If the no. of characters in the string are NOT in multiples of 3, and if there are two characters more than multiple of 3, then the FRONT and END parts will get one extra character each, as seen in the example of “Johny”
STEP TWO: Concatenate (join) the FRONT, MIDDLE and END parts of the strings as per the below specified concatenation-rule to form three Output strings.
Output1 = FRONT part of Input1 + FRONT part of Input2 + FRONT part of Input3
Output2 = MIDDLE part of Input1 + MIDDLE part of Input2 + MIDDLE part of Input3
Output3 = END part of Input1 + END part of Input2 + END part of Input3
For example, for the above specified example input strings,
Output1 = “J” + “Jo” + “Jan” = “JJoJan”
Output2 = “oh” + “h” + “ard” = “ohhard”
Output3 = “n” + “ny” + “han” = “nnyhan”
Step THREE: After the first two steps, we will have three output strings. Further processing is required only for the third output string as per below rule –
“ Toggle the case of each character in the string”, i.e. in the third output string, all lower-case characters should be made upper-case and vice versa.
For example, for the above example strings, Output3 is “nnyhan”, so after applying the toggle rule, Output3 should become “NNYHAN”.
Final Result – The three output strings after applying the above three steps is the final result. i.e. for the above example,
Output1 = “JJoJan”
Output2 = “ohhard”
Output3 = “NNYHAN”
Anand approaches you to help him write a program that would do the above mentioned processing on any given three strings and generate the resulting three output strings
Note that the three output strings should be returned as members of a "Result" object/struct.

```
import java.io.*;
import java.util.*;
// Read only region start
class UserMainCode {

    public class Result {
        public final String output1;
        public final String output2;
        public final String output3;
```

```
        public Result(String out1, String out2, String out3) {
            output1 = out1;
            output2 = out2;
            output3 = out3;
        }
    }

    public Result encodeThreeStrings(String input1, String
input2, String input3) {

        String array1[] = new String[3];
        String array2[] = new String[3];
        String array3[] = new String[3];

        array1 = getparts(input1);
        array2 = getparts(input2);
        array3 = getparts(input3);

        String output1 = "";
        String output2 = "";

        output1 = array1[0] + array2[0] + array3[0];
        output2 = array1[1] + array2[1] + array3[1];
        StringBuilder output3 = new StringBuilder(array1[2] +
array2[2] + array3[2]);

        for (int i = 0; i < output3.length(); i++) {
            if (Character.isUpperCase(output3.charAt(i))) {
                output3.setCharAt(i,
Character.toLowerCase(output3.charAt(i)));
            } else {
                output3.setCharAt(i,
Character.toUpperCase(output3.charAt(i)));
            }
        }
```

```
        return new Result(output1, output2,
output3.toString());

    }

    public static String[] getparts(String inputs) {
        int len = inputs.length();
        String parts[] = new String[3];
        int partLen = len / 3;
        // case-1: 9/3=3
        // case-2: 4/3=1
        // case-3: 5/3=1

        System.out.println("str : " + inputs + "Len : " + len +
"partLen : " + partLen);

        if (len % 3 == 0) {
            parts[0] = inputs.substring(0, partLen);
            parts[1] = inputs.substring(partLen, 2 * partLen);
            parts[2] = inputs.substring(2 * partLen, len);
            System.out.println("For case-1 :" + parts[0] + " "
+ parts[1] + " " + parts[2]);
        }
        if (len % 3 == 1) {
            parts[0] = inputs.substring(0, partLen);
            parts[1] = inputs.substring(partLen, 2 * partLen +
1);
            parts[2] = inputs.substring(2 * partLen + 1, len);
            System.out.println("For case-2 :" + parts[0] + " "
+ parts[1] + " " + parts[2]);
        }
        if (len % 3 == 2) {
            parts[0] = inputs.substring(0, partLen + 1);
            parts[1] = inputs.substring(partLen + 1, 2 *
partLen + 1);
            parts[2] = inputs.substring(2 * partLen + 1, len);
```

```
            System.out.println("For case-2 :" + parts[0] + " "
+ parts[1] + " " + parts[2]);
        }
        return parts;
    }
}
public class EncodingThreeStrings {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String input1=sc.nextLine();
        String input2=sc.nextLine();
        String input3=sc.nextLine();
        UserMainCode um=new UserMainCode();
        System.out.println("OutPut1 : "+
um.encodeThreeStrings(input1, input2, input3).output1);
        System.out.println("OutPut2 :
"+um.encodeThreeStrings(input1, input2, input3).output2);
        System.out.println("OutPut3 :
"+um.encodeThreeStrings(input1, input2, input3).output3);
    }
}
```

Milestone-3(logic Building Questions)

Generate series and find Nth element:

Given three numbers, a, b and c, such that
either
a < b < c
or
a > b > c
and
a, b, and c can be positive, negative or zero.
Form the series such that the gap between c and its next element (say d) should be the same as the gap between a and b. Similarly, the gap between c's next element (d) and d's next element (say e) should be the same as the gap between b and c.
Find and return the Nth element.
Example1- If the three numbers are a=1, b=3, c=6 and N=17
The series will be formed as below –
1, 3, 6, 8, 11, 13, 16, 18, 21, 23, 26, 28, 31, 33, 36, 38, 41
17 th number in the series is 41
Example2- If the three numbers are a=5, b=-2, c=-4 and N=14
The series will be formed as below –
5, -2, -4, -11, -13, -20, -22, -29, -31, -38, -40, -47, -49, -56
14 th number in the series is -56
The function prototype should be as below –
int seriesN(int a, int b, int c, int N);

```
import java.util.Scanner;

public class GenerateSeriesAndFindNthElement{
    public static int findPassword(int input1,int input2,int
input3,int input4)
    {
        int gapAB=input2-input1;
        int gapBC=input3-input2;
        int output=input1;
        for(int i=1;i<input4;i++)
        {
```

```
            if(i%2==1)
                output+=gapAB;
            else
                output+=gapBC;
        }
        return output;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();
        int N=sc.nextInt();
        System.out.println(findPassword(a, b, c, N));
    }
}
```

Find result after alternate add sub on N:

Given a number N (1=N<=10000), and an option opt=1 or 2, find the re below rules, If opt=1," Result= N-(N-1) + (N-2) (N-3) + (N-4)... till 1 If opt=2.

Result= N+ (N-1)-(N-2)+ (N-3)-(N-4)... till 1

Example1: If N = 6, and opt=1 Result = 6-54-3+2-1=3

Example2: If N = 6, and opt=2

Result = 6+5-4+3-2+1=9

The function prototype should be as below

int AddSub(int N, int opt):

```java
import java.util.Scanner;

public class FindResultAfterAlternateAdd_SubOnN {
    public static int findResultAfterAlternateAdd_SubOnN(int
input1,int input2)
    {
        int flag;
        int output=input1;
        if(input2==2)
            flag=1;
        else
            flag=0;

        for(int i=input1-1;i>=1;i--)
        {
            if(flag==1)
            {
                output+=i;
                flag=0;
            }
            else
            {
                output-=i;
                flag=1;
            }
        }
        return output;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int input1=sc.nextInt();
        int option=sc.nextInt();

System.out.println(findResultAfterAlternateAdd_SubOnN(input1,
option));
    }
```

```
}
```

Find Password:

Detective Buckshee Junior has been approached by the shantiniketan kids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee Junior realises that he will need a programmer's support. He contacts you and requests your help. Please help the detective by writing a function to generate the password.

The scenario is as below

Five numbers are available with the kids.

These numbers are either stable or unstable

A number is stable if each of its digit occur the same number of times, ie, the frequency of each digit in the number is the same. For eg 2277, 4004, 11, 23, 583835, 1010 are examples of stable numbers. Similarly. A number is unstable if the frequency of each digit in the number is NOT the same For eg 221, 4314, 101, 233, 58135, 101 are examples of unstable numbers.

The password can be found as below. Le password = sum of all stable numbers- sum of all unstable numbers.

Assuming that the five numbers are passed to a function as input1, input2, input3. input4 and input5, complete the function to find and return the password

For Example:

If input1= 12, input2= 1313, input3= 122, input4= 678, and input5 = 898, stable numbers are 12, 1313 and 678 unstable numbers are 122 and 898

So, the password should be = sum of all stable numbers- sum of all unstable numbers 983

```java
import java.util.ArrayList;

class UserMainCode
{
    public int findPassword(int input1,int input2,int
input3,int input4,int input5)
    {
        int stableSum=0,unstableSum=0;
        ArrayList<Integer>arr=new ArrayList<>();
        arr.add(input1);
        arr.add(input2);
        arr.add(input3);
        arr.add(input4);
        arr.add(input5);

        for(int i=0;i<arr.size();i++)
        {
            if(isStable(arr.get(i)))
            {
                stableSum+=arr.get(i);
            }
            else
            {
                unstableSum+=arr.get(i);
            }
        }
        return (stableSum-unstableSum);
    }
    public static boolean isStable(int num)
    {
        int arr[]=new int [10];
        int p=-1;
        while (num>0) {
            arr[num%10]++;
            num/=10;
        }
```

```
        for(int i=0;i<10;i++)
        {
            if(arr[i]!=0 && p==-1)
            {
                p=arr[i];
            }
            else if(arr[i]!=0 && p!=arr[i])
            {
                return false;
            }
        }
        return true;
    }
}
public class Findpassword_Stable_Unstable
{
    public static void main(String[] args) {
        UserMainCode um=new UserMainCode();
        System.out.println(um.findPassword(12, 1313, 122, 678,
898));
    }
}
```

calculate sum of non-prime index values in an array:

what is a prime number? a prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. in other words, a prime number is a whole number greater than 1, whose only two whole-number factors are 1 and itself. the first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29. given an array with 'n' elements, you are expected to find the sum of the values that are present in non-prime indexes of the array. note that the array index starts with 0 i.e. the position (index) of the first array element is 0, the position of the next array element is 1. and so on. example 1 if

the array elements are (10, 20, 30, 40, 50, 60, 70, 80, 90, 100), then the values at the non-prime index are 10,20,50,70,90,100 and their sum is 340. example 2: if the array elements are (-1, -2, -3, 3, 4.-7), then the values at the non-prime index are-1,-2.4 and their sum is 1. example 3. if the array elements are (-4,-2), the values at the non-prime index are 4, 2 and their sum is -6 the function prototype should be as below int sumofnonprimeindexvaluesfint input[], int input2); where input1 is the given array, and input2 is the no. of elements in the array.

```
import java.util.ArrayList;
import java.util.Scanner;

public class SumOfNonPrimeIndexvalues{
    public static int sumOfNonPrimeIndexvalues(int []input1,int
input2)
    {
        int sum=0;
        ArrayList<Integer>arr=new ArrayList<>();
        for(int i=0;i<input2;i++)
            if(isPrime(i)==false)
                arr.add(input1[i]);
        for( int i=0;i<arr.size();i++)
            sum+=arr.get(i);
        return sum;
    }
    public static boolean isPrime(int n)
    {
        if(n<=1)
            return false;
        for(int i=2;i<n;i++)
            if(n%i==0)
                return false;
        return true;
    }
    public static void main(String[] args) {
```

```
        Scanner sc=new Scanner(System.in);
        int array[]={};
        int size=array.length;

        System.out.println(sumOfNonPrimeIndexvalues(array,
size));
    }
}
```

Find the one digit to be removed to form palindrome –

Assume that the given number input1 can become a palindrome if only one of its digit is removed. i.e. only one digit in the number is out of place. Find the digit that needs to be removed from input1 to convert input1 to a palindrome.
Example1: If input1 = 12332, the digit '1' needs to be removed to convert input1 to a palindrome 2332. So, the function should return 1.
Example2: If input1 = 251532, the digit '3' needs to be removed to convert input1 to a palindrome 25152. So, the function should return 3.
Example3: If input1 = 10101, NO digit needs to be removed to convert input1 to a palindrome, because it is already a palindrome. So we must return -1 in this case.
Example4: If input1 = 981894, In the digit '4' needs to be removed to convert input1 to a palindrome 98189. So, the function should return 4.

```
import java.io.*;
import java.util.*;

class FindTheOneDigitToBeRemovedToFormPalindrome {

    public int digitRemove_Palin(int input1){
        StringBuilder num = new
StringBuilder(String.valueOf(input1));
        for (int i = 0; i < num.length(); i++) {
```

```
            if (palindromeCheck(num.toString())) return -1;

            char removedChar = num.charAt(i);
            String newNum = num.deleteCharAt(i).toString();

            if (palindromeCheck(newNum)) {
                System.out.println(i + ":: " + newNum + " :: "
+ removedChar);
                return
Integer.parseInt(String.valueOf(removedChar));
            } else {
                num.insert(i, removedChar);
            }
        }
        return -1;
    }
    public static boolean palindromeCheck(String input1) {
        input1 = input1.toLowerCase();
        int digitCount = input1.length();
        boolean isPalindrome = true;

        int range = digitCount / 2;
        if (digitCount % 2 == 0) range--;

        for (int i = 0; i <= range; i++) {
            if (input1.charAt(i) != input1.charAt(digitCount -
i - 1)) isPalindrome = false;
        }
        return isPalindrome;
    }
}
```

The “Nambiar Number” Generator:

M.Nambiar has devised a mechanism to process any given mobile number and thus generate a new resultant number. He calls this mechanism as the “Nambiar Number Generator” and the resultant number is referred to as the “Nambiar Number”. The mechanism is as follows “ In the given mobile number, starting with the first digit, keep on adding all subsequent digits till the state (even or odd) of the sum of the digits is opposite to the state (odd or even) of the first digit. Continue this from the subsequent digit till the last digit of the mobile number is reached. Concatenating the sums thus generated results in the Nambiar Number.”
The below examples will help to illustrate this.
Please also look at the bottom of this problem description for the expected function prototype.
Example 1
If the given mobile number is 9880127431
The first pass should start with the first digit.
First digit is 9 which is odd.
So, we will keep adding subsequent digits till the sum becomes even.
9+8=17 (17 is odd, so continue adding the digits)
9+8+8 = 25 (25 is odd, so continue adding the digits)
9+8+8+0 = 25 (25 is odd, so continue adding the digits)
9+8+8+0+1 = 26 (26 is even, which is opposite to the state of the first digit 9)
So, Stop first pass here and remember that the result at the end of first pass = 26
Now we enter the second pass.
The second pass should start after the digit where we stopped the first pass.
In the first pass we have added the digits 9,8,8,0 and 1.
So, the first digit for second pass will be 2, which is even.
Now, we will keep adding subsequent digits till the sum becomes odd.
2+7=9 (9 is odd, which is opposite to the state of the first digit 2)
So, Stop second pass here and remember that the result at the end of second pass = 9
Now we enter the third pass.
In the first pass we have added the digits 9,8,8,0 and 1, and the resultant sum was 26.
In the second pass we have added the digits 2 and 7, and the resultant sum was 9.
The third pass should start after the digit where we stopped the second pass.
So, the first digit for third pass will be 4, which is even.
Now, we will keep adding subsequent digits till the sum becomes odd.
4+3=7 (7 is odd, which is opposite to the state of the first digit 4)
So, Stop third pass here and remember that the result at the end of third pass = 7
Now we enter the fourth pass.
In the first pass we have added the digits 9,8,8,0 and 1, and the resultant sum was 26.
In the second pass we have added the digits 2 and 7, and the resultant sum was 9.

In the third pass we have added the digits 4 and 3, and the resultant sum was 7.
The fourth pass should start after the digit where we stopped the third pass.
So, the first digit for fourth pass will be 1, which is odd.
Now, we will keep adding subsequent digits till the sum becomes even.
However, we realize that this digit 1 is the last digit of the mobile number and there are no further digits. So, Stop fourth pass here and remember that the result at the end of fourth pass = 1

For the mobile number 9880127431, the resultant number (Nambiar Number) will be the concatenation of the results of the 4 passes = [26][9][7][1] = 26971
Note1: Please note that the number of passes required to process the given number may vary depending upon the constitution of the mobile number.
Note2: Also note that, 0 should be considered as an even number
Example 2
If the given mobile number is 9860857152
First digit 9 is odd.
First pass results in 9+8+6+0+8+5= 36
Second pass results in 7+1= 8
Third pass results in 5+2= 7
Note that the third pass stops at 7 (even though we do not meet a change of state) because we have reached the end of the mobile number.

For the mobile number 9860857152, the resultant number (Nambiar Number) will be the concatenation of the results of the 3 passes = [36][8][7] = 3687

Example 3
If the given mobile number is 8123454210
The resultant number (Nambiar Number) will be 95970
Example 4
If the given mobile number is 9900114279
The resultant number (Nambiar Number) will be 181149

```
import java.util.Scanner;

public class TheNamiarNumberGenerator {
    public static int nnGenerator(String input1)
    {
        String mobileNo=input1;
```

```
        StringBuilder numbiarNo=new StringBuilder();

        for(int i=0;i<mobileNo.length();i++)
        {
            int
firstDigit=Integer.parseInt(String.valueOf(mobileNo.charAt(i)))
;

            int firstDigitEvenOrOdd=firstDigit%2==0?0:1;
            int sum=firstDigit;
            int  j=i+1;

            if(j==mobileNo.length())
            {
                numbiarNo.append(firstDigit);
                break;
            }

            while (true) {

sum+=Integer.parseInt(String.valueOf(mobileNo.charAt(j++)));
        if(sum%2!=firstDigitEvenOrOdd||j>=mobileNo.length())
                {
                    numbiarNo.append(sum);
                    i=j-1;
                    break;
                }
            }
        }
        return Integer.parseInt(numbiarNo.toString());
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String mobileNo=sc.nextLine();
        int output=nnGenerator(mobileNo);
        System.out.println(output);
```

```
    }
}
```

User ID Generation:

Joseph's team has been assigned the task of creating user-ids for all participants of an online gaming competition. Joseph has designed a process for generating the user-id using the participant's First Name, Last Name PIN code and a number N. The process defined by Joseph is as below

Step1-Compare the lengths of First Name and Last Name of the participant. The one that is shorter will be called "Smaller Name" and the one that is longer will be called the "Longer Name" If both First Name and Last Name are of equal Length, then the name that appears earlier in alphabetical order will be called "Smaller Name" and the name that appears later in alphabetical order will be called the "Longer Name".

Step2 - The user-id should be generated as below Last Letter of the smaller name Entire word of the longer name + Digit at position N in the PIN when traversing the PIN from left to right+Digit at position N in the PIN when traversing the PIN from right to left

Step3- Toggle the alphabets of the user-id generated in step-2 i.e. upper-case alphabets should become lower-case and lower-case alphabets should become upper case

Let us see a few examples

Example-1- If the participant's details are as below

First Name Rajiv Last Name = Roy

PIN-560037

N=6

Step1-Length of Last Name is less than the Length of First Name, so the Smaller

Name is "Roy" and the Longer Name is "Rajiv

Step2 - The user-id will be Last Letter of the smaller name + Entire word in the longer name + Digit at position N in the PIN when traversing the PIN from left to right + Digit at position N in the PIN when traversing the PIN from right to left Last Letter of "Roy" + Entire word in "Rajiv + 6th Digit of PIN from left+ 6th Digit of

PIN from right = y + Rajiv +7 +5

Therefore, user-id=yRajiv75

Step3 Toggle the alphabets in the user-id. So, user-id=YrAJIV75

Example-2- If the participant's details are as below

First Name Manoj Last Name Kumar

PIN=561327

N=2

Step1-Length of First Name is equal to the Length of Last Name Alphabetically.

"Kumar' appears earlier than Manoj (by comparing alphabetic positions of K and 'M) so the Smaller Name is "Kumar and the Longer Name is "Manoj"

Step2 - The user-id will be = Last Letter of the smaller name Entire word in the longer name + Digit at position N in the PIN when traversing the PIN from left to right + Digit at position N in the PIN when traversing the PIN from right to left = Last Letter of "Kumar+ Entire word in "Manoj" + 2nd Digit of PIN from left + 2nd

Digit of PIN from right

=r+ Manoj + 6+2

Therefore, user-id = rManoj62

Step3 Toggle the alphabets in the user-id. So, user id= RmANOJ62

Example-3-If the participant's details are as below First Name Kumud == Last Name Kumar

PIN 561327

N=2

Step1-Length of First Name is equal to the Length of Last Name. Alphabetically.

Kumar appears earlier than Kumud (by comparing alphabetic positions of 'Kuma a "Kumu) so the Smaller Name is "Kumar" and the Longer Name is "Kumud

Step2 - The user-id will be = Last Letter of the smaller name + Entire word in the longer name + Digit at position N in the PIN when traversing the PIN from left to right + Digit at position N in the PIN when traversing the PIN from right to left - Last Letter of "Kumar" + Entire word in "Kumud - 2nd Digit of PIN from Digit of PIN from right

=r+Kumud +6+2

Therefore, user-id=rKumud62

Step3 - Toggle the alphabets in the user-id. So, user-id= RkUMUD62

You are part of Joseph's team and he has asked you to write a program (method) to generate the participant's user-id using the above rules.

You are expected to write the logic within the method

(function) userldGeneration which provides 4 inputs as below

input1 is the First Name,

input2 is the Last Name

input3 is the PIN

input4 is the number N

The method (function) should do the processing as per rules explained above and should return the generated user-id.

Assumption For convenience of this assessment question, Let us assume that the value of N (input4) will always be less than or equal to the number of digits in the PIN.

```
import java.io.*;
import  java.util.*;

// Read only region start
class Program1
{
    public static String userIdGeneration(String input1,String
input2,int input3,int input4){

        String smallerName,largerName;
        String pin=Integer.toString(input3);
        if(input1.length()>input2.length())
        {
            largerName=input1;
            smallerName=input2;
        }
        else if(input1.length()<input2.length())
```

```
        {
            largerName=input2;
            smallerName=input1;
        }
        else
        {
            int compare=input1.compareTo(input2);
            if(compare<0)
            {
                largerName=input2;
                smallerName=input1;
            }
            else
            {
                largerName=input1;
                smallerName=input2;
            }
        }

        String userId=smallerName.charAt(smallerName.length()-
1)+largerName+pin.charAt(input4-1)+pin.charAt(pin.length()-
input4);
        StringBuffer sb=new StringBuffer(userId);
        for(int i=0;i<userId.length();i++)
        {
            if(Character.isLowerCase(userId.charAt(i)))
            {
                sb.setCharAt(i,
Character.toUpperCase(userId.charAt(i)));
            }
            else
            {
                sb.setCharAt(i,
Character.toLowerCase(userId.charAt(i)));
            }
        }
```

```
        return sb.toString();
    }
    public static void main(String[] args) {

        System.out.println(userIdGeneration("Debasis", "Dutta",
561327, 6));
    }
}
```

Message controlled Robot movement with 90 degrees turning capability and 1 unit moving capability

Harish, an engineering student needs to submit his final year project. He decides to create a Robot which can be controlled by a set of instructions. He also decides that a grid (of X and Y axis) should be defined and the robot should move only within that grid. The set of instructions to move the robot should be given as a single message (string) and the Robot should accordingly move and reach the expected location. If the given instructions lead to a position which is out of the given grid, the Robot should stop at the last valid instruction.
Harish decides to write a function named moveRobot that should process the given inputs and return a string representing the final position of the Robot. The function moveRobot will take 4 input parameters that define the size of the grid (X and Y axis), the current position of the Robot, and the message (string) containing the set of movement instructions.
The first two input parameters define the size of the grid. input1 = X axis of the grid input2 = Y axis of the grid Note that input1 and input2 will always be > 0. So, the valid grid area for the robot's movement should be the rectangular area formed between the diagonal ends (0,0) and (X,Y)
The third parameter defines the current (starting) position of the robot.
input3 = current position of the robot, represented as a string containing 3 values separated by a – (hyphen). The format of input3 is x-y-D, where x and y represent the current (starting) position of the robot and D represents the direction where the robot is currently facing. Valid values for direction D are E, W, N, or S, representing East, West, North and South respectively.
The fourth input parameter represents the single message containing the set of instructions to move the robot.

input4 = movement Instructions to the robot, represented as a string containing the instructions separated by a space. The message will consist of a series of M, L or R, where
M means "Move 1 unit forward in the direction that the robot is facing",
L means "Turn 90 0 towards left", and
R means "Turn 90 0 towards right".
Output expected to be returned by the function -
The function is expected to process the given inputs and return a string representing the final position of the Robot. The returned string should be of the format x-y-D, where x and y represent the final (end) position of the robot and D represents the direction where the robot is finally facing. Valid values for direction D are E, W, N, or S, representing East, West, North and South respectively. Note that an " -ER" must be appended to the output string if the traversal finally stopped due to an invalid move (see the below two examples for more clarity on this)
Note:
You can assume the grid to be similar to the 1st quadrant of the regular graph sheet. In a regular graph sheet of dimensions x units and y units, (0,0) is the bottom left corner and (x,y) is the top right corner. Ex: For a grid of 5 x 5, the bottom left corner will be (0,0) and top right corner will be (5,5).
The starting position of the robot (third input parameter) will be any position on the grid. i.e. it need not always be (0,0)
You can assume that the current position (starting position, specified in input3) will always be a valid position within the specified grid.
IMPORTANT - Note that the instructions L and R only change the direction of the robot without moving it. The instruction M moves the robot 1 unit forward in the direction that the robot is facing.
Invalid moves should not be allowed - Any move that could lead the robot to a position beyond (outside) the defined x and y axis of the grid OR below 0 on either x or y axis, should be considered an invalid move. (see below examples to get clarity)
Example1 –
input1: x = 3
input2: y = 3
input3: 2-2-E
input4: R M L M L M
Output: The function should return 3-2-N
Explanation:
The size of the grid is 3x3 units. Current (starting) position of the robot is (2,2) facing East. After processing the set of instructions given in input4, the new position will be in (3,2) facing North. So, the function is expected to return the output in the format x-y-D i.e. 3-2-N
Example2 –
input1: x = 3
input2: y = 4

input3: 2-2-E
input4: R M L M L M R M
Output: The function should return 3-2-E-ER
Explanation:
The size of the grid is 3x4 units. Current (starting) position of the robot is (2,2) facing East. After processing the set of instructions given in input4, the new position will be in (3,2) facing East. Note that the last instruction (M) leads to a position outside the grid, so the valid moves stop at R which is the second last instruction. In this case, the function is expected to return the output representing the last valid position appended with "–ER" representing ERROR. So, the function should return the output as 3-2-E-ER
IMPORTANT NOTE: The output format should be strictly as specified above. Any extra spaces before, after or within the output string will result in failure. Also, the alphabets in the output string should be in upper-case.
NOTE: The above few examples are only to help you understand the question. The actual test-case values will be different from these, so you must ensure to check the result for all possible cases.

```
import java.io.*;
import  java.util.*;

// Read only region start
class MessageControlledRobotMovement {

    public String moveRobot(int input1,int input2,String
input3,String input4){
        // Read only region end
        int X = input1;
        int Y = input2;
        String currentPos = input3;
        String msg = input4;
        int currX = Integer.parseInt(currentPos.split("-")[0]);
        int currY = Integer.parseInt(currentPos.split("-")[1]);
        String currD = currentPos.split("-")[2]; // E/W/N/S
        String[] instructions = msg.split(" "); // M L R M M L
M ...
        StringBuilder output = new StringBuilder();
```

```
        System.out.println(Arrays.toString(instructions));
        System.out.println("Curr: " + currX + currY + currD);
        for (int i = 0; i < instructions.length; i++) {
            System.out.print(instructions[i] + ":: ");
            if (instructions[i].equals("M")) {
                if (currD.equals("E") && (currX + 1 > X )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("W") && (currX - 1 < 0 )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("N") && (currY + 1 > Y )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("S") && (currY - 1 < 0 )) {
                    output.append("-ER");
                    break;
                }
                if (currD.equals("E")) currX++;
                else if (currD.equals("W")) currX--;
                else if (currD.equals("N")) currY++;
                else if (currD.equals("S")) currY--;
            } else {
                if (currD.equals("E") &&
instructions[i].equals("L"))
                    currD = "N";
                else if (currD.equals("E") &&
instructions[i].equals("R"))
                    currD = "S";
                else if (currD.equals("W") &&
instructions[i].equals("L"))
                    currD = "S";
```

```
                else if (currD.equals("W") &&
instructions[i].equals("R"))
                    currD = "N";
                else if (currD.equals("N") &&
instructions[i].equals("L"))
                    currD = "W";
                else if (currD.equals("N") &&
instructions[i].equals("R"))
                    currD = "E";
                else if (currD.equals("S") &&
instructions[i].equals("L"))
                    currD = "E";
                else if (currD.equals("S") &&
instructions[i].equals("R"))
                    currD = "W";
            }
            output.delete(0, output.length());
            output.append(currX + "-" + currY + "-" + currD);
            System.out.println(output);
        }
        return output.toString();
    }
}
```