1) IDENTIFY POSSIBLE WORDS:

Identify possible words: Detective Bakshi while solving a case stumbled upon a letter which had many words whose one character was missing ie. 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 **identity PossibleWords** 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 then return the final **output** string in the format specified below

See below examples carefully to understand the format of **input1, input2** and **output**

Example1:

input1= "Fi_er”

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

output= "FILER:FIXER FIBER”

Note that-

- 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 be “ERROR-009”.

Code For _ :

```
import java.util.regex.*;
int i,j;
input1 = input1.toUpperCase();
input2 = input2.toUpperCase();
String word="", res="";
String[] words = input2.split(":");
for(i=0;i<words.length;i++)
{
   word = words[i];
   if(input1.length()==word.length())
   {
      for(j=0;j<input1.length();j++)
      {
         if(input1.charAt(j)!='_' && input1.charAt(j)!=word.charAt(j))
         {
            break;
         }
      }
      if(j==input1.length())
      {
         res=res + word+":";
      }
   }
}
return res.length()==0?"ERROR-009":res.substring(0,res.length()-1);
```

CODE FOR _ and ?:

```
import java.util.regex.*;

    input1 = input1.toUpperCase();

    input2 = input2.toUpperCase();

    String[] words = input2.split(":");

    String word="", res="";

    for(int i=0;i<input1.length();i++)

    {

     if(input1.charAt(i)=='_')

        word = word + "[A-Z]";

     else if(input1.charAt(i)=='?')

        word = word + "[A-Z]?";

     else

        word = word + input1.charAt(i);

    }
```

```
for(int i=0;i<words.length;i++)

{

   if(Pattern.matches(word, words[i]))

     res = res + words[i]+":";

}

return res.length()==0?"ERROR-009":res.substring(0,res.length()-1);
```

2) ALTERNATE ADD SUM

Given a number N (1<=N<=10000), and an option opt=1 or 2, find the result as per 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: IfN = 6, and opt=1
Result =6-5+4-3+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);

Code:

```
int sum=0;
sum+=N;
 N=N-1;
 if(opt==2)
 {
    if(N%2==0)
    {
       while(N>0)
       {
          if(N%2==0)
          sum+=N;
          else
          sum-=N;
          N--;
       }
    }
    else
    {
       while(N>0)
       {
```

```
            if(N%2==0)
            sum-=N;
            else
            sum+=N;
            N--;
        }
    }
}
else
{
    if(N%2==0)
    {
        while(N>0)
        {
            if(N%2==0)
            sum-=N;
            else
            sum+=N;
            N--;
        }
    }
    else
    {
        while(N>0)
        {
            if(N%2==0)
            sum+=N;
            else
            sum-=N;
            N--;
        }
    }
}
return sum;
```

3) 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 difference of its original value and its succeeding element's value.

i.e. arr[i] = arr[i+1]-arr[i]

e.g. value in arr[0] = original value of arr[1] - original value of arr[0]

Also note that value of last element i.e. arr[last index] remains unchanged.

Example:If the original array is –{2, 5, 1, 7, 9, 3}

The encoded array would be –{3, -4, 6, 2, -6, 3}

Provided the encoded array, you are expected to find the –a) First number (value in index 0) in the original arrayb) Sum of all numbers in the original array

The prototype of the function is:public static void findOriginalFirstAndSum(int[] input1,input 2); where input1 is the encoded array.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.

Assumption(s):

- The array elements can be positive and/or negative numbers

Example 1:Original array = {2, 5, 1, 7, 9, 3} Encoded array = {3, -4, 6, 2, -6, 3} First number in original array = 2 Sum of all numbers in original array = 27

Code:

```
int[] res = new int[input1.length];

res[res.length - 1] = input1[input1.length - 1];

for (int i = input1.length - 1; i > 0; i--)

{

    res[i - 1] = input1[i - 1] - res[i];

}

int sum = 0;

for (int item : res)

    sum += item;

return new Result(res[0], sum);
```

4) FIND SUMEET SUM (Model-1)

Given 5 input numbers, Sumeet has to find the sum of the smallest numbers that can be produced using 2 digits from each of the above 5 numbers.

Understanding Question:

We are given five numbers and we have to find the smallest 2-digit numbers from given five numbers i.e., For example take a number 54110. We have to find the smallest 2-digit number that can be formed. In the first Iteration we will get 0 as the smallest number when considered as 1-digit number but we need 2-digit number so in another iteration we'll get 1 as smallest number so from this the smallest 2-digit number that can be formed from given number 54110 is 01.

In the similar manner we're given 5 numbers and we have to find the smallest 2-digit number from all those inputs given and we've to find the sum of all smallest 2-digit numbers acquired.

Explaining Given Test Case 1:
Input1 – 23792 – From these the smallest number that can be formed is 22.
Input2 – 37221 – From these the smallest number that can be formed is 12.
Input3 – 10270 – From these the smallest number that can be formed is 00.
Input4 – 73391 – From these the smallest number that can be formed is 13.
Input5 – 12005 – From these the smallest number that can be formed is 00.
The second part of our task is to find the sum of all these smallest 2-digit numbers and the result is 47.

Explaining Given Test Case 2:
Input1 – 26674 – From these the smallest number that can be formed is 24.
Input2 – 105 – From these the smallest number that can be formed is 01.
Input3 – 37943 – From these the smallest number that can be formed is 33.
Input4 – 95278 – From these the smallest number that can be formed is 25.
Input5 – 27845 – From these the smallest number that can be formed is 24.
The second part of our task is to find sum of all these smallest 2-digit numbers and the result is 107.

Code:

```
    int sum=0;
    char ch1[]=(String.valueOf(input1)).toCharArray();
    Arrays.sort(ch1);

    sum+=Integer.parseInt(String.valueOf(ch1[0])+String.valueOf(ch1[1]));

    char ch2[]=(String.valueOf(input2)).toCharArray();

    Arrays.sort(ch2);

    sum+=Integer.parseInt(String.valueOf(ch2[0])+String.valueOf(ch2[1]));

    char ch3[]=(String.valueOf(input3)).toCharArray();

     Arrays.sort(ch3);
sum+=Integer.parseInt(String.valueOf(ch3[0])+String.valueOf(ch3[1]));

char ch4[]=(String.valueOf(input4)).toCharArray();
 Arrays.sort(ch4);
sum+=Integer.parseInt(String.valueOf(ch4[0])+String.valueOf(ch4[1]));

char ch5[]=(String.valueOf(input5)).toCharArray();
Arrays.sort(ch5);
sum+=Integer.parseInt(String.valueOf(ch5[0])+String.valueOf(ch5[1]));

return sum;
```

5) FIND SUMEET SUM (Model-2)

FindSumeetSum: Sum of smallest 3- digit numbers from given 5 numbers
Given 5 input numbers, Sumeet has to find the sum of the smallest numbers that can be produced using 3 digits from each of the above 5 numbers

Example-1
If the 5 input numbers are 23792,37221,10270,73391 and 12005
The smallest numbers that can be produced using 3 digits from each of these are 223,122,001,133 and 001 respectively, and the sum of these smallest numbers will be 480.
Therefore, the expected result is 480

Example-2
If the 5 input numbers are 26674,105,37943,95278 and 27845,
The smallest numbers that can be produced using 3 digits from each of these are 246,015,334,257 and 245 respectively, and the sum of these smallest numbers will be 1097.
Therefore, the expected result is 1097.
NOTE- All the given 5 numbers will be >=100 and <=99999
Function prototype is as below-
Int findSumeetSum(int input1,int input2,int input3,int input4,int input5)

Code:

```
int sum=0;
char ch1[]=(String.valueOf(input1)).toCharArray();
Arrays.sort(ch1);
sum+=Integer.parseInt(String.valueOf(ch1[0]) + String.valueOf(ch1[1]) + String.valueOf(ch1[2]));
char ch2[]=(String.valueOf(input2)).toCharArray();
Arrays.sort(ch2);
sum+=Integer.parseInt(String.valueOf(ch2[0]) + String.valueOf(ch2[1]) + String.valueOf(ch2[2]));
char ch3[]=(String.valueOf(input3)).toCharArray();
Arrays.sort(ch3);
sum+=Integer.parseInt(String.valueOf(ch3[0]) + String.valueOf(ch3[1]) +  String.valueOf(ch3[2]));
char ch4[]=(String.valueOf(input4)).toCharArray();
Arrays.sort(ch4);
sum+=Integer.parseInt(String.valueOf(ch4[0]) + String.valueOf(ch4[1]) + String.valueOf(ch4[2]));
char ch5[]=(String.valueOf(input5)).toCharArray();
Arrays.sort(ch5);
sum+=Integer.parseInt(String.valueOf(ch5[0]) + String.valueOf(ch5[1]) + String.valueOf(ch5[2]));
return sum;
```

6) FIND SUMEET SUM (Model-3)

FindSumeetSum: Sum of largest 3-digit numbers from given 5 numbers
Given 5 input numbers, Sumeet has to find the sum of the largest numbers that can be produced using 3 digits from each of the above 5 numbers

Example-1
If the 5 input numbers are 23792,37221,10270,73391 and 12005,
The largest numbers that can be produced using 3 digits from each of these are
973,732,721,973 and 521 respectively and the sum of these largest numbers will be 3920.
Therefore, the expected result is 3920.

Example-2
If the 5 input numbers are 26674,105,37943,95278 and 27845
The largest numbers that can be produced using 3 digits from each of these are
766,510,974,987 and 875 respectively and the sum of these largest numbers will be 4112.
Therefore, the expected result is 4112.

NOTE:- All the given 5 numbers will be >=100 and <=99999
Function prototype is as below –
Int findSumeetSum(int input1,int input2,int input3,int input4,int input5)

Code:

```
int sum=0,l;
char ch1[]=(String.valueOf(input1)).toCharArray();
Arrays.sort(ch1);
l=ch1.length;
sum+=Integer.parseInt(String.valueOf(ch1[l-1]) + String.valueOf(ch1[l-2]) + String.valueOf(ch1[l-3]));
char ch2[]=(String.valueOf(input2)).toCharArray();
Arrays.sort(ch2);
l=ch2.length;
sum+=Integer.parseInt(String.valueOf(ch2[l-1]) + String.valueOf(ch2[l-2]) + String.valueOf(ch2[l-3]));
char ch3[]=(String.valueOf(input3)).toCharArray();
Arrays.sort(ch3);
l=ch3.length;
sum+=Integer.parseInt(String.valueOf(ch3[l-1]) + String.valueOf(ch3[l-2]) + String.valueOf(ch3[l-3]));
char ch4[]=(String.valueOf(input4)).toCharArray();
Arrays.sort(ch4);
l=ch4.length;
sum+=Integer.parseInt(String.valueOf(ch4[l-1]) + String.valueOf(ch4[l-2]) + String.valueOf(ch4[l-3]));
char ch5[]=(String.valueOf(input5)).toCharArray();
Arrays.sort(ch5);
l=ch5.length;
sum+=Integer.parseInt(String.valueOf(ch5[l-1]) + String.valueOf(ch5[l-2]) + String.valueOf(ch5[l-3]));
return sum;
```

7) FIND SUMEET SUM (Model-4)

Find SumeetSum: Sum of smallest 2-digit numbers from given 4 numbers
Given 4 input numbers, Sumeet has to find the sum of the smallest numbers that can be produced using 2 of each of the above 4 numbers

Example-1
If the 4 input numbers are 23792,37221,10270 and 73391,
The smallest numbers that can be produced using 2 digits from each of these are 22,12,00 and 13 respectively the sum of these smallest numbers will be 47. Therefore, the expected result is 47.

Example-2
If the 4 input numbers are 26674,105,37943 and 95278,
The smallest numbers that can be produced using 2 digits from each of these are 24,01,33 and 25 respectively the sum of these smallest numbers will be 83. Therefore, the expected result is 83.

NOTE- All the given 4 numbers will be >=100 and <=99999
Function prototype is as below-
Int findSumeetSum(int input1,int input2,int input3,int input4)

Code:

```
int sum=0;
char ch1[]=(String.valueOf(input1)).toCharArray();
Arrays.sort(ch1);
sum+=Integer.parseInt(String.valueOf(ch1[0])+String.valueOf(ch1[1]));
char ch2[]=(String.valueOf(input2)).toCharArray();
Arrays.sort(ch2);
sum+=Integer.parseInt(String.valueOf(ch2[0])+String.valueOf(ch2[1]));
char ch3[]=(String.valueOf(input3)).toCharArray();
Arrays.sort(ch3);
sum+=Integer.parseInt(String.valueOf(ch3[0])+String.valueOf(ch3[1]));
char ch4[]=(String.valueOf(input4)).toCharArray();
Arrays.sort(ch4);
sum+=Integer.parseInt(String.valueOf(ch4[0])+String.valueOf(ch4[1]));
return sum;
```

8) ENCODED TWO STRINGS (Model-1)

You are provided with TWO words. You are expected to split each word into THREE parts each, and create a password using the below rule –

Password = first part of word 2 + first part of word1 + third part of word1 + third part of word 2

For splitting a given word into three parts the below approach should be used
If word= "ABC", then part1-A part2=B and part3-C
If word= "ABCD", then part1=A part2=BC and part 3=D
If word= "ABCDE" then part1=A part2=BCD and part3=E

If word= "ABCDEF", then part1=AB part2=CD and part3=EF
If word= "ABCDEFG", then part1 AB part2-DCDE and part3-FG
If word "ABCDEFGH", then part1 AB part2-COEF and part3=GH
and so on
i.e.,

1) If the length of the given word can be equally divided into three parts, then each part gets the same number of letters (as seen in above examples of "ABC" and "ABCDEF")
2) If the length of the given word cannot be equally divided into three parts, then the center part i.e., part2 gets the extra number of characters (as seen in rest of the above examples)

Let us now look at the below examples

Example 1: input1="WIPRO" input 2="TECHNOLOGIES"
Output should be "TECHWOGIES"
Explanation -
The three parts of WIPRO will be "W", "IPR" and "O"
The three parts of TECHNOLOGIES will be "TECH", "NOLO" and "GIES"
So,
Password = First part of word 2 + First part of word 1+ Third part of word 1 + Third part of word2
=TECH + W + O + GIES
= TECHWOGIES

Example 2: input="MACHINE" input 2="LEARNING"
Output should be "LEMANENG"
Explanation -
The three parts of MACHINE will be "MA", "CHI" and "NE"
The three parts of LEARNING will be "LE", "ARNI", and "NG"
So,
Password = First part of word 2 + First part of word 1+ Third part of word 1 + Third part of word2
= LE + MA + NE + NG
= LEMANENG

Code:

```
String[][] res = new String[2][3];
String[] words = {input1, input2};
String password="";
for(int i=0;i<2;i++)
{
   int l = words[i].length();
   res[i][0] = words[i].substring(0,l/3);
   res[i][1] = words[i].substring(l/3, l-l/3);
   res[i][2] = words[i].substring(l-l/3);
```

```
}
password = res[1][0] + res[0][0] + res[0][2] + res[1][2];
return password;
```

9) ENCODED TWO STRINGS (Model-2)

You are provided with TWO words. You are expected to split each word into THREE parts each, and create a password using the below rule –

Password = Second part of word 1+ Second part of word 2 + First part of word 2

For splitting a given word into three parts the below approach should be used
If word= "ABC", then part1-A part2=B and part3-C
If word= "ABCD", then part1=A part2=BC and part 3=D
If word= "ABCDE" then part1=A part2=BCD and part3=E
If word= "ABCDEF", then part1=AB part2=CD and part3=EF
If word= "ABCDEFG", then part1 AB part2-DCDE and part3-FG
If word "ABCDEFGH", then part1 AB part2-COEF and part3=GH
and so on
i.e.,

1) If the length of the given word can be equally divided into three parts, then each part gets the same number of letters (as seen in above examples of "ABC" and "ABCDEF")
2) If the length of the given word cannot be equally divided into three parts, then the center part i.e., part2 gets the extra number of characters (as seen in rest of the above examples)

Let us now look at the below examples

Example 1: input1="WIPRO" input 2="TECHNOLOGIES"
Output should be "IPRNOLOTECH"
Explanation -
The three parts of WIPRO will be "W", "IPR" and "O"
The three parts of TECHNOLOGIES will be "TECH", "NOLO" and "GIES"
So,
Password = Second part of word 1+ Second part of word 2 + First part of word 2
= IPR + NOLO + TECH
= IPRNOLOTECH
Example 2: input="MACHINE" input 2="LEARNING"
Output should be "CHIARNILE"
Explanation -
The three parts of MACHINE will be "MA", "CHI" and "NE"
The three parts of LEARNING will be "LE", "ARNI", and "NG"
So,
Password = Second part of word 1+ Second part of word 2 + First part of word 2
= LE + MA + NE + NG
= LEMANENG

Code:

```
String[][] res = new String[2][3];
String[] words = {input1, input2};
String password="";
for(int i=0;i<2;i++)
{
   int l = words[i].length();
   res[i][0] = words[i].substring(0,l/3);
   res[i][1] = words[i].substring(l/3, l-l/3);
   res[i][2] = words[i].substring(l-l/3);
}
password = res[0][1] + res[1][1] + res[1][0];
return password;
```

10) ENCODED TWO STRINGS (Model-3)

You are provided with TWO words. You are expected to split each word into THREE parts each, and create a password using the below rule –

Password = Second part of word1+ Third part of word 2 + First part of word1 + First part of word 2

For splitting a given word into three parts the below approach should be used
If word= "ABC" then part1=A, part2=B and part3=C
if word= "ABCD" then part1=A, part2=BC and part3=D
if word= "ABCDE" then part1=AB, part2=C and part3=DE
if word= "ABCDEF" then part1=AB, part2=CD and part3=EF
if word= "ABCDEFG" then part1=AB, part2=CDE and part3=FG
if word = "ABCDEFGH" then part1=ABC, part2=DE and part3=FGH
and so on
i.e.,

1) If the length of the given word can be equally divided into three parts, then each part gets the same number letters (as seen in above examples of "ABC" and "ABCDEF")
2) If after dividing the length of the given word into three parts, there is one extra character left, then the extra Character goes to the middle part i.e., part2. (as seen in above examples of "ABCD" and "ABCDEFG")
3) If after dividing the length of the given word into three parts, there are two extra characters left, then part1 and part3 get the extra characters (as seen in above examples of "ABCDE" and "ABCDEFGH")

Let us now look at the below examples

Example 1: input1="WIPRO" input 2="TECHNOLOGIES"
Output should be "PGIESWITECH"
Explanation -
The three parts of WIPRO will be "WI", "P" and "RO"
The three parts of TECHNOLOGIES will be "TECH", "NOLO" and "GIES"
So,

Password = Second part of word1+ Third part of word 2 + First part of word1 + First part of word 2
=P + GIES + WI + TECH
= PGIESWITECH

Example 2: input="MACHINE" input 2="LEARNING"
Output should be "CHIINGMALEA"
Explanation -
The three parts of MACHINE will be "MA", "CHI" and "NE"
The three parts of LEARNING will be "LEA", "RN", and "ING"
So,
Password = Second part of word1+ Third part of word 2 + First part of word1 + First part of word 2
= CHI + ING + MA + LEA
= CHIINGMALEA

Code:

```
String[] words = {input1, input2};
String[][] parts = new String[2][3];
for(int i=0;i<2;i++)
{
   int len = words[i].length();
   if(len%3==0 || len%3==1)
   {
       parts[i][0] = words[i].substring(0, len/3);
       parts[i][1] = words[i].substring(len/3, len-len/3);
       parts[i][2] = words[i].substring(len-len/3);
    }
   else
   {
      parts[i][0] = words[i].substring(0, len/3+1);
      parts[i][1] = words[i].substring(len/3+1, len-len/3-1);
      parts[i][2] = words[i].substring(len-len/3-1);
   }
}
return (parts[0][1]+parts[1][2]+parts[0][0]+parts[1][0]);
```

11) ENCODED TWO STRINGS (Model-4)

You are provided with TWO words. You are expected to split each word into THREE parts each, and create a password using the below rule –

Password = Third part of word2 + Second part of word1 + Second part of word2 + First part of word1
For splitting a given word into three parts the below approach should be used
If word= "ABC" then part1=A, part2=B and part3=C
if word= "ABCD" then part1=A, part2=BC and part3=D
if word= "ABCDE" then part1=AB, part2=C and part3=DE
if word= "ABCDEF" then part1=AB, part2=CD and part3=EF

if word= "ABCDEFG" then part1=AB, part2=CDE and part3=FG
if word = "ABCDEFGH" then part1=ABC, part2=DE and part3=FGH
and so on
i.e.,

1) If the length of the given word can be equally divided into three parts, then each part gets the same number letters (as seen in above examples of "ABC" and "ABCDEF")
2) If after dividing the length of the given word into three parts, there is one extra character left, then the extra Character goes to the middle part i.e., part2. (as seen in above examples of "ABCD" and "ABCDEFG")
3) If after dividing the length of the given word into three parts, there are two extra characters left, then part1 and part3 get the extra characters (as seen in above examples of "ABCDE" and "ABCDEFGH")

Let us now look at the below examples

Example 1: input1="WIPRO" input 2="TECHNOLOGIES"
Output should be "GIESPNOLOWI"
Explanation -
The three parts of WIPRO will be "WI", "P" and "RO"
The three parts of TECHNOLOGIES will be "TECH", "NOLO" and "GIES"
So,
Password = Third part of word2 + Second part of word1 + Second part of word2 + First part of word1
= GIES + P + NOLO + WI
= GIESPNOLOWI

Example 2: input="MACHINE" input 2="LEARNING"
Output should be "INGCHIRNMA"
Explanation -
The three parts of MACHINE will be "MA", "CHI" and "NE"
The three parts of LEARNING will be "LEA", "RN", and "ING"
So,
Password = Third part of word2 + Second part of word1 + Second part of word2 + First part of word1
= ING + CHI + RN + MA
= INGCHIRNMA

Code:

```
String[] words = {input1, input2};
String[][] parts = new String[2][3];
for(int i=0;i<2;i++)
{
   int len = words[i].length();
   if(len%3==0 || len%3==1)
   {
      parts[i][0] = words[i].substring(0, len/3);
      parts[i][1] = words[i].substring(len/3, len-len/3);
      parts[i][2] = words[i].substring(len-len/3);
   }
```

```
        else
        {
            parts[i][0] = words[i].substring(0, len/3+1);
            parts[i][1] = words[i].substring(len/3+1, len-len/3-1);
            parts[i][2] = words[i].substring(len-len/3-1);
        }
    }
    return (parts[1][2]+parts[0][1]+parts[1][1]+parts[0][0]);
```

12) ENCODED TWO STRINGS (Model-5)

You are provided with TWO words. You are expected to split each word into THREE parts each, and create a password using the below rule –

Password = First part of word2 + First part of word1 + Third part of word1 + Third part of word2

For splitting a given word into three parts the below approach should be used
If word= "ABC" then part1=A, part2=B and part3=C
if word= "ABCD" then part1=A, part2=BC and part3=D
if word= "ABCDE" then part1=AB, part2=C and part3=DE
if word= "ABCDEF" then part1=AB, part2=CD and part3=EF
if word= "ABCDEFG" then part1=AB, part2=CDE and part3=FG
if word = "ABCDEFGH" then part1=ABC, part2=DE and part3=FGH
and so on
i.e.,

1) If the length of the given word can be equally divided into three parts, then each part gets the same number letters (as seen in above examples of "ABC" and "ABCDEF")
2) If after dividing the length of the given word into three parts, there is one extra character left, then the extra Character goes to the middle part i.e., part2. (as seen in above examples of "ABCD" and "ABCDEFG")
3) If after dividing the length of the given word into three parts, there are two extra characters left, then part1 and part3 get the extra characters (as seen in above examples of "ABCDE" and "ABCDEFGH")

Let us now look at the below examples
Example 1: input1="WIPRO" input 2="TECHNOLOGIES"
Output should be "TECHWIROGIES"
Explanation -
The three parts of WIPRO will be "WI", "P" and "RO"
The three parts of TECHNOLOGIES will be "TECH", "NOLO" and "GIES"
So,
Password = First part of word2 + First part of word1 + Third part of word1 + Third part of word2
= TECH + WI + RO + GIES
= TECHWIROGIES

Example 2: input="MACHINE" input 2="LEARNING"

Output should be "LEAMANEING"
Explanation -
The three parts of MACHINE will be "MA", "CHI" and "NE"
The three parts of LEARNING will be "LEA", "RN", and "ING"
So,
Password = First part of word2 + First part of word1 + Third part of word1 + Third part of word2
= LEA + MA + NE + ING
= LEAMANEING

Code:

```
String[] words = {input1, input2};
String[][] parts = new String[2][3];
for(int i=0;i<2;i++)
{
   int len = words[i].length();
   if(len%3==0 || len%3==1)
   {
      parts[i][0] = words[i].substring(0, len/3);
      parts[i][1] = words[i].substring(len/3, len-len/3);
      parts[i][2] = words[i].substring(len-len/3);
   }
   else
   {
      parts[i][0] = words[i].substring(0, len/3+1);
      parts[i][1] = words[i].substring(len/3+1, len-len/3-1);
      parts[i][2] = words[i].substring(len-len/3-1);
   }
}
return (parts[1][0]+parts[0][0]+parts[0][2]+parts[1][2]);
```

13) FIND KEY (Model-1):

Find Key:

You are provided with 3 numbers : input1, input2 and input3.

Each of these are 4 digits numbers within >=1000 and <=9999

i.e.,

1000<=input1<=9999

1000<=input2<=9999

1000<=input3<=9999

You are expected to find the key using below formula:

Key = Sum of Largest digits of each number + Sum of Second Largest digits of each number

For Example, input1=3521, input2=2452 input3=1352

Key = (5+5+5) + (3+4+3) = 25

Assuming three numbers are passed to given function and complete the function to find and return the Key

ANSWER:

```
int th1=input1/1000;
int h1=(input1/100)%10;
int t1=(input1/10)%10;
int u1=input1%10;
int []a={h1,t1,u1,th1};
Arrays.sort(a);
int l1=a[3];
int sl1=a[2];
int th2=input2/1000;
int h2=(input2/100)%10;
int t2=(input2/10)%10;
int u2=input2%10;
int []b={h2,t2,u2,th2};
Arrays.sort(b);
int l2=b[3];
int sl2=b[2];
int th3=input3/1000;
int h3=(input3/100)%10;
int t3=(input3/10)%10;
int u3=input3%10;
int []c={h3,t3,u3,th3};
Arrays.sort(c);
```

```
int l3=c[3];
int sl3=c[2];
int key=(l1+l2+l3)+(sl1+sl2+sl3);
return key;
```

14) FIND KEY (Model-2):

You are provided with 3 numbers input1,input2,input3.

Each of these are four digit numbers within the range >=1000 and <=9999

i.e

1000<=input1<=9999

1000<=input2<=9999

1000<=input3<=9999

you are expected to find the key using the below formula

Key=[smallest digit in the thousands place of all three numbers][LARGEST digit in the hundreds place of all the three numbers]

[smallest digit in the tens place of all three numbers][LARGEST digit in the units place of all three numbers]

for e.g if input1=3521,input2=2452,input3=1352,then Key=[1][5][2][2]=1522

Assuming that the 3 numbers are passed to the given function.Complete the function to find and return the key.

ANSWER:

```
int th1=input1/1000;
int h1=(input1/100)%10;
int t1=(input1/10)%10;
int u1=input1%10;

int th2=input2/1000;
int h2=(input2/100)%10;
```

```
int t2=(input2/10)%10;+
int u2=input2%10;


 int th3=input3/1000;
 int h3=(input3/100)%10;
 int t3=(input3/10)%10;
  int u3=input3%10;


    int mth=Math.min(Math.min(th1,th2),th3);
    int mh=Math.max(Math.max(h1,h2),h3);
    int mt=Math.min(Math.min(t1,t2),t3);
    int mu=Math.max(Math.max(u1,u2),u3);


    int key=(mth*1000)+(mh*100)+(mt*10)+mu;
    return key;
```

15) FIND KEY (Model-3):

You are provided with 3 numbers input1,input2,input3.

Each of these are four digit numbers within the range >=1000 and <=9999//

1000<=input1<=9999
1000<=input2<=9999
1000<=input3<=9999
you are expected to find the key using the below formula

Key=[LARGEST digit in the thousands place of all three numbers][smallest digit in the hundreds place of all the three numbers][LARGEST digit in the tens place of all three numbers][smallest digit in the units place of all three numbers]

for e.g if input1=3521,input2=2452,input3=1352,then Key=[3][3][5][1]=3351

Assuming that the 3 numbers are passed to the given function.Complete the function to find and return the key.

ANSWER:

```
 int th1=input1/1000;
int h1=(input1/100)%10;
```

```
int t1=(input1/10)%10;
int u1=input1%10;

 int th2=input2/1000;
int h2=(input2/100)%10;
int t2=(input2/10)%10;
int u2=input2%10;

 int th3=input3/1000;
int h3=(input3/100)%10;
int t3=(input3/10)%10;
int u3=input3%10;

int mth=Math.max(Math.max(th1,th2),th3);
int mh=Math.min(Math.min(h1,h2),h3);
int mt=Math.max(Math.max(t1,t2),t3);
int mu=Math.min(Math.min(u1,u2),u3);

int key=(mth*1000)+(mh*100)+(mt*10)+mu;
return key;
```

16) TRAVERSE ARRAY AND FIND KEY (Model-1)

Question

Traverse Array and Find Key –

Mohan has received an array of numbers

The numbers in this array are special because each number consists of two parts -a “KEY” part and a “NEXT ADDRESS” part For example, if the number in the array is 411, the leftmost digit in the number is “4” is the “KEY part and all the remaining digits in number be “11 form the “NEXT ADDRESS part.

Mohan’s task is to start from the first array element, pick the “KEY”part go to the “NEXT ADDRESS array element pick Its “KEY” part, go to the “NEXT ADDRESS array element, pick its “KEY part, and continue this cycle the encounters a negative number While traversing through the array in this fashion, we need to perform an alternate addition and subtraction of

the KEYS.

The result of alternate addition and subtraction of all the keys is the expected final result. Note that we should stop traversing (traveling) through the array when a negative number is encountered (See Examples 1 and 2 below) Important: If the array does NOT contain any negative number, the result should be the largest number in the array (See Example 3 below)

Help Mohan by writing the code to find the FINAL Result. Input1 represents the array of numbers, and input2 represents the number of elements in the array.

Example 1 –

If the array input 1 is 74 -56 15 71 92 23 and input2 is 6

First array element = 74

Here, KEY = 7

NEXT_ADDRESS – 4

4th array element = 92 (NOTE THAT ARRAY ELEMENT ADDRESS STARTS FROM 0, SO 4th element is 92)

Here, KEY = 9 NEXT ADDRESS = 2

2nd array element 15

Here, KEY = 1, NEXT ADDRESS = 5

5th array element – 23

Here KEY = 2 NEXT ADDRESS = 3

3rd array element 71

Here, KEY 7, NEXT ADDRESS =1

1st array element =-56

Here, KEY = 5 NEXT ADDRESS -STOP (because we have reached a negative number).

FINAL RESULT = Alternatively Add and Subtract the keys = 7+9-1+ 2-7+5= 15

Code:

```
int i, flag=0;
int max = Integer.MIN_VALUE;
for(i=0;i<input2;i++)
{
   if(input1[i]<0)
      flag=1;
   if(input1[i]>max)
      max=input1[i];
}
```

```
if(flag==0)
   return max;

int sign=-1, res=0, key, digits, power;
i=0;
while(i<input2 && flag==1)
{
   if(input1[i]<0)
   {
      flag=0;
      input1[i]*=-1;
   }
   digits = (int)(Math.log10(input1[i])+1);
   power = (int)(Math.pow(10, digits-1));
   key = (int)input1[i]/power;
   if(i==0)
      res = key;
   else
      res = res + key*(sign=sign*-1);
   i = input1[i]%power;
}
return res;
```

17) TRAVERSE ARRAY AND FIND KEY (Model-2)

Mohan has received an array of numbers

The numbers in this array are special because each number consists of two parts -a “KEY” part and a “NEXT ADDRESS” part For example, if the number in the array is 411, the leftmost digit in the number le “11” is the “KEY part and all the remaining digits in number be “4 form the “NEXT ADDRESS part.

Mohan’s task is to start from the first array element, pick the “KEY”part go to the “NEXT ADDRESS array element pick Its “KEY” part, go to the “NEXT ADDRESS array element, pick its “KEY part, and continue this cycle the encounters a negative number While traversing through the array in this fashion, ne needs to perform an alternate addition and subtraction of the KEYS.

The result of alternate addition and subtraction of all the keys is the expected final result. Note that we should stop traversing (traveling) through the array when a negative number is encountered (See

Examples 1 and 2 below) Important: If the array does NOT contain any negative number, the resume should be the largest number in the array.

Note that we should stop traversing (traveling)through the array when a negative number is encountered (See Examples 1 and 2 below)

Important: If the array does NOT contain any negative number, the result should be the smallest number in the array

(See Example 3 below)

Help Mohan by writing the code to find the FINAL Result.

Input represents the array of numbers, and input represents the number of elements in the array

Example 1-1 the array input1 is{ 47,-65,51,17,29,32} and input2 is 6

First array element 47

Here, KEY 7,NEXT ADDRESS = 4

4th array element 29 (NOTE THAT ARRAY ELEMENT ADDRESS STARTS FROM 0, So 4th element is 29)

Here, KEY 9 NEXT ADDRESS= 2

2nd array element = 51

Here, KEY 1 NEXT ADDRESS = 5

5th array element = 32

Here, KEY 2 NEXT ADDRESS=3

3rd array element 17

Here KEY 7 NEXT ADDRESS = 1

1st array element = -65

Here, KEY=5 NEXT ADDRESS - STOP (because we have reached a negative number)

FINAL RESULT = Alternately subhead and Add the key 7-9+1-2+7-5=1

Example 2 - If the array s {47,65,51,17,29,-32} and input2 is 6

First array element 47

Here KEY 7 NEXT ADDRESS=4

4th array element is 29

Here, KEY 9 NEXT ADDRESS =2

2nd may element is 51

Here KEY 1 NEXT ADDRESS=5

5th array element - 32

Here, KEY- 2 NEXT_ADDRESS-STOP (because we have reached a negative number)

FINAL RESULT Animatedly Subtract and Add the key =7-9+1-2=-3

Example 3 - the array is 47 65 51 12 29 32 54 and input2 is 7

Here we see that the array does NOT contain any negative number, so the result should be calculated as the smallest number in the array

FINAL RESULT 12

ANSWER:

```
import java.util.*;

public class MyClass {

   public int findKey(int input1[],int input2){

      int flag=0,k=0,key,add;

      int a[]=new int[input2];

for(int i=0;i<input2;i++){

         if(input1[i]<0){

            flag=1;

break;

         }

      }

      if(flag==0){

Arrays.sort(input1);

         return input1[0];

      }

else{

         key=input1[0]%10;

         a[k++]=key;

         input1[0]=input1[0]/10;
```

```
        add=input1[0];
        while(input1[add]>0){
           key=input1[add]%10;
           a[k++]=key;
           input1[add]=input1[add]/10;
           add=input1[add];
        }
        a[k++]=-(input1[add]%10);
     }
     int sum=0;
for(int i=0;i<k;i++){
        if(i%2==0)
        sum+=a[i];
        else
        sum-=a[i];
     }
     return sum;
  }
```

18) FIND PASSWORD (Model-1)

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 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 digits occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **password=(Number of unstable numbers*10)+Number of stable numbers**

Assuming that the fiVE numbers are passed to a function as input1,input2,input3,input4,input 5.complete the function to find and return the password.

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers i.e 12,1313 and 678 and

TWO unstable numbers i.e 122 and 898

so,the password should be=(Number of Unstable numbers*10)+Number of stable numbers=(2*10)+3=23

****

```
int[] num = {input1, input2, input3, input4, input5};

int stable=0, unstable=0, i, j;

for(i=0;i<5;i++)

{

   int[] freq = new int[10];  //frequencies of all the digits

   int temp=num[i];

   int maxf=0;

   while(temp!=0)

   {
```

```
        int r = temp%10;
        freq[r]++;
        temp/=10;
        if(freq[r]>maxf)
            maxf=freq[r];
    }
    for(j=0;j<10;j++)
    {
        if(freq[j]!=0 && freq[j]!=maxf)
            break;
    }
    if(j==10)
        stable++;
    else
        unstable++;
}
return (unstable*10 + stable);
```

19) FIND PASSWORD (Model-2)

Detective Buckshee junior has been approached by the shantiniketankids 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 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g:2277,4004,11,23,583835,1010 are examples of stable numbers.

Similarily ,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-

i.e **password=(Number of stable numbers*10)+Number of unstable numbers**

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

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers i.e 12,1313 and 678 and

TWO unstable numbers i.e 122 and 898

so,the password should be=(Number of stable numbers*10)+Number of Unstable numbers=(3*10)+1=32

ANSWER:

```
int[] num = {input1, input2, input3, input4, input5};

int stable=0, unstable=0, i, j;

for(i=0;i<5;i++)

{

  int[] freq = new int[10];  //  frequencies of all the digits
```

```
  int temp=num[i];
  int maxf=0;
  while(temp!=0)
  {
      int r = temp%10;
      freq[r]++;
      temp/=10;
      if(freq[r]>maxf)
          maxf=freq[r];
  }
  for(j=0;j<10;j++)
  {
      if(freq[j]!=0 && freq[j]!=maxf)
          break;
  }
  if(j==10)
      stable++;
  else
      unstable++;
}
return (stable*10 + unstable);
```

20) FIND PASSWORD (Model-3)

Detective Buckshee junior has been approached by the shantiniketankids 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 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 digits occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **Password=Maximum of all stable numbers+Minimum of all Unstable numbers.**

Assuming that the five numbers are passed to a function as input1,input2,input3,input4,input 5.complete the function to find and return the password.

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers i.e 12,1313 and 678 and

TWO unstable numbers i.e 122 and 898

so,the password should be=Maximum of all stable numbers+Minimum of all Unstable numbers=1313+122=1435

ANSWER:

```
int input[]=new int[]{input1,input2,input3,input4,input5};

        int h[]=new int[10];

        String s="";

        int st=0,u=0;
```

```
        int min=Integer.MAX_VALUE;
        int max=Integer.MIN_VALUE;
for(int i=0;i<input.length;i++){
for(int j=0;j<10;j++)
            h[j]=0;
            s=String.valueOf(input[i]);
for(int j=0;j<s.length();j++){
                h[Integer.parseInt(s.substring(j,j+1))]++;
            }
            int c=0,k=-1,flag=0;
for(int j=0;j<10;j++){
                if(h[j]!=0 && c==0){
                    k=h[j];
c++;
                }
                if(h[j]!=0 && h[j]!=k){
                    flag=1;
break;
                }
            }
            if(flag==1)
            {
                if(input[i]<min)
                min=input[i];
            }
            else
            {
```

```
            if(input[i]>max)
            max=input[i];
        }
    }
    return max+min;
```

21) FIND PASSWORD (Model-4)

Detective Buckshee junior has been approached by the shantiniketankids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee junior realizes that he will need a programmer's support. He contacts you 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **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, input5. Complete the function to find and return the password.

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers 12,1313 and 678 and

TWO unstable numbers 122 and 898

So, the password should be=sum of all stable numbers – sum of all Unstable numbers=983

ANSWER:

```
int[] num = {input1, input2, input3, input4, input5};

int stable=0, unstable=0, i, j;

for(i=0;i<5;i++)

{

   int[] freq = new int[10];  //  frequencies of all the digits

   int temp=num[i];

   int maxf=0;

   while(temp!=0)

   {

       int r = temp%10;

       freq[r]++;

       temp/=10;

       if(freq[r]>maxf)

           maxf=freq[r];

   }

   for(j=0;j<10;j++)

   {

       if(freq[j]!=0 &&freq[j]!=maxf)

           break;

   }
```

```
    if(j==10)
        stable=stable+num[i];
    else
       unstable=unstable+num[i];
}
return (stable - unstable);
```

22) FIND PASSWORD (Model-5)

Detective Buckshee junior has been approached by the shantiniketankids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee junior realizes that he will need a programmer's support. He contacts you 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **Password=Maximum of all stable numbers - Minimum of all Unstable numbers**

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

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers 12,1313 and 678 and

TWO unstable numbers 122 and 898

So, the Password should be=Maximum of all stable numbers - Minimum of all Unstable numbers=1313-122=1191

ANSWER:

```
int input[]=new int[]{input1,input2,input3,input4,input5};

        int h[]=new int[10];

        String s="";

        int st=0,u=0;

        int min=Integer.MAX_VALUE;

        int max=Integer.MIN_VALUE;

for(int i=0;i<input.length;i++){

for(int j=0;j<10;j++)

            h[j]=0;

            s=String.valueOf(input[i]);

for(int j=0;j<s.length();j++){

                h[Integer.parseInt(s.substring(j,j+1))]++;

            }

            int c=0,k=-1,flag=0;

for(int j=0;j<10;j++){

                if(h[j]!=0 && c==0){

                    k=h[j];

c++;

                }

                if(h[j]!=0 && h[j]!=k){

                    flag=1;

break;
```

```
            }
        }
        if(flag==1)
        {
            if(input[i]<min)
            min=input[i];
        }
        else
        {
            if(input[i]>max)
            max=input[i];
        }
    }
    return max-min;
```

---

23) FIND PASSWORD (Model-6)

Detective Buckshee junior has been approached by the shantiniketankids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee junior realizes that he will need a programmer's support. He contacts you 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **Password=Maximum of all Unstable numbers - Minimum of all Unstable numbers**

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

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers 12,1313 and 678 and

TWO unstable numbers 122 and 898

So, the Password should be=Maximum of all Unstable numbers - Minimum of all Unstable numbers=898-122=776

ANSWER:

```
int input[]=new int[]{input1,input2,input3,input4,input5};

        int h[]=new int[10];

        String s="";

        int l=0;

        int res[]=new int[input.length];

for(int i=0;i<input.length;i++){

for(int j=0;j<10;j++)
```

```
            h[j]=0;
            s=String.valueOf(input[i]);
for(int j=0;j<s.length();j++){
                h[Integer.parseInt(s.substring(j,j+1))]++;
            }
            int c=0,k=-1,flag=0;
for(int j=0;j<10;j++){
                if(h[j]!=0 && c==0){
                    k=h[j];
c++;
                }
                if(h[j]!=0 && h[j]!=k){
                    flag=1;
break;
                }
            }
            if(flag==1)
            res[l++]=input[i];
        }
        int min=Integer.MAX_VALUE,max=Integer.MIN_VALUE;
for(int i=0;i<l;i++)
        {
            if(res[i]<min)
            min=res[i];
            if(res[i]>max)
            max=res[i];
        }
```

```
        return max-min;
```

24) FIND PASSWORD (Model-7)

Detective Buckshee junior has been approached by the shantiniketankids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee junior realizes that he will need a programmer's support. He contacts you 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **Password=sum of all Unstable numbers**

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

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers 12,1313 and 678 and

TWO unstable numbers 122 and 898

So, the Password should be=sum of all Unstable numbers=898+122=1020

ANSWER:

```
int[] num = {input1, input2, input3, input4, input5};
int stable=0, unstable=0, i, j;
for(i=0;i<5;i++)
{
  int[] freq = new int[10];  //  frequencies of all the digits
  int temp=num[i];
  int maxf=0;
  while(temp!=0)
  {
      int r = temp%10;
      freq[r]++;
      temp/=10;
      if(freq[r]>maxf)
          maxf=freq[r];
  }
  for(j=0;j<10;j++)
  {
      if(freq[j]!=0 &&freq[j]!=maxf)
          break;
  }
  if(j==10)
      stable=stable+num[i];
  else
     unstable=unstable+num[i];
}
return unstable;
```

25) FIND PASSWORD (Model-8)

Detective Buckshee junior has been approached by the shantiniketankids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee junior realizes that he will need a programmer's support. He contacts you 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **Password=Maximum of all stable numbers - Minimum of all stable numbers**

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

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers 12,1313 and 678 and

TWO unstable numbers 122 and 898

So, the Password should be=Maximum of all stable numbers - Minimum of all stable numbers=1313-12=1301

ANSWER:

```
int input[]=new int[]{input1,input2,input3,input4,input5};
        int h[]=new int[10];
        String s="";
        int l=0;
        int res[]=new int[input.length];
for(int i=0;i<input.length;i++){
for(int j=0;j<10;j++)
            h[j]=0;
            s=String.valueOf(input[i]);
for(int j=0;j<s.length();j++){
                h[Integer.parseInt(s.substring(j,j+1))]++;
            }
            int c=0,k=-1,flag=0;
for(int j=0;j<10;j++){
                if(h[j]!=0 && c==0){
                    k=h[j];
c++;
                }
                if(h[j]!=0 && h[j]!=k){
                    flag=1;
break;
                }
            }
            if(flag==0)
            res[l++]=input[i];
        }
```

```
        int min=Integer.MAX_VALUE,max=Integer.MIN_VALUE;
for(int i=0;i<l;i++)
        {
            if(res[i]<min)
            min=res[i];
            if(res[i]>max)
            max=res[i];
        }
        return max-min;
```

---

26) FIND PASSWORD (Model-9)

Detective Buckshee junior has been approached by the shantiniketankids society for help in finding the password to the games complex. After hearing the scenario, detective Buckshee junior realizes that he will need a programmer's support. He contacts you 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 numbers is stable if each of its digit occur the same number of times, i.e the frequency of each digit in the number is the same.

For e.g: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-

i.e **Password=Maximum of all stable numbers + Minimum of all stable numbers**

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

For example:

If input1=12,input2=1313,input3=122,input4=678 and input5=898 , we see that there are THREE stable numbers 12,1313 and 678 and

TWO unstable numbers 122 and 898

So, the Password should be=Maximum of all stable numbers + Minimum of all stable numbers=1313 + 12=1325

ANSWER:

```
int input[]=new int[]{input1,input2,input3,input4,input5};
        int h[]=new int[10];
        String s="";
        int l=0;
        int res[]=new int[input.length];
for(int i=0;i<input.length;i++){
for(int j=0;j<10;j++)
            h[j]=0;
            s=String.valueOf(input[i]);
for(int j=0;j<s.length();j++){
                h[Integer.parseInt(s.substring(j,j+1))]++;
            }
            int c=0,k=-1,flag=0;
for(int j=0;j<10;j++){
```

```
                if(h[j]!=0 && c==0){
                    k=h[j];
c++;
                }
                if(h[j]!=0 && h[j]!=k){
                    flag=1;
break;
                }
            }
            if(flag==0)
            res[l++]=input[i];
        }
        int min=Integer.MAX_VALUE,max=Integer.MIN_VALUE;
for(int i=0;i<l;i++)
        {
            if(res[i]<min)
            min=res[i];
            if(res[i]>max)
            max=res[i];
        }
        return max+min;
```

27) PERSONS AND TOKENS (Model-1)

There are N people sitting in a room

Each of them have been given a token number Only 3 people out of them have their token numbers in

a sequence

You are expected to find the 3 people whose numbers are in sequence in this context, sequence means numbers which are in continuous order for eg 97, 98,99 are numbers in a sequence However, 3, 5, 7 should not Be considered a continuous sequence.

You are expected to write the logic within a given function find Sequence which takes 3 input parameters - Input 1= number, representing the number of elements in the array - input2 and input3 (This also represents,N the number of people in the room)

Input 2[] = array of strings representing the names of the people sitting in the room

Input3[] = array of integers representing the token number of the people sitting in the room Note:- There is a one-to-one mapping between the array input 3 and input 2, ie.

input 3[0] represents the token number of the person input 2[0]

input 3[1] represents the token number of the person input 2[1]

input 3[2] represents the token number of the person input 2[2]

and so on

Expected output: The function find Sequence is expected to return a string containing the names of the 3 persons whose token numbers are in sequence . Note that the names of the 3 persons should be in descending order of their token numbers. The names should be separated using a colon. If there is no sequence available in the given input then return " NONE", find the examples below.

Example 1:

Input1 = 10

Input2 = {“Rajesh", “ Abdul", “Rahul”, “Priya”, “Sanjay", "Nidhi”, “Varun”, “Varsha”, “Basil”, “Asif”}

Input3 = {99, 46, 39, 102, 45, 521, 65, 4, 47, 741}

Expected output "Basil:Abdul: Sanjay"

Explanation: Out of the 10 token numbers, the numbers 45, 46 and 47 are in sequence and their descending order 47 46 and 45 Hence the expected output is a string containing names separated by a colon of the 3 persons who hold these tokens. Note that the names are in descending sequence of their respective taken numbers,

Example 2:

Input1= 7

Input = ["aa", "bb", "cc", "dd", "ee", "gg"]

Inpul 3={9,89,5,0,6,65,4}

Expected output = "ee:cc:gg"

Code:

```
int i,x=0,y=0,z=0,flag=0;
int[] a=input3.clone();
Arrays.sort(a);
for(i=0;i<input1-2;i++)
 {
    if(a[i+1]-a[i]==1 && a[i+2]-a[i+1]==1)
    {
            flag=1;
            x=a[i+2];
            y=a[i+1];
            z=a[i];
            break;
     }
 }
 if(flag==0)
return "NONE";

int fi=0,si=0,ti=0;
for(i=0;i<input1;i++)
 {
    if(x==input3[i])
            fi=i;
    if(y==input3[i])
            si=i;
    if(z==input3[i])
            ti=i;
}
return input2[fi]+":"+input2[si]+":"+input2[ti];
```

28) PERSONS AND TOKENS (Model-2)

Input2[] = array of strings representing the names of the people sitting in the room

Input3[] = array of integers representing the token number of the people sitting in the room

Note: There is a one-to-one mapping between the array input3 and input2, i.e.

Input3[0] represents the token number of the person input2[0]

Input3[1] represents the token number of the person input2[2]

Input3[2] represents the token number of the person input 2[2]

……..and so on

Expected output: The function findsequence is expected to return a **string** containing the **names of the 3 persons** whose token numbers are in sequence. Note that the names of the 3 persons should be **in ascending order** of their token numbers. The names should be separated using a colon. If there is no sequence available in the given input then return “NONE”, find the examples below

Example 1:

Input1 = 10

Input2 = {“Rajesh","Abdul”, "Rahul”, "Priya","Sanjay", "Nidhi", “Varun" ,”Varsha”, “Basil", “Asif”)

Input3 = {99,46,39,102,45,521,65,4,47,741}

Expected output = "Sanjay:Abdul:Basil"

Explanation: Out of the 10 token numbers, the numbers 45, 46 and 47 are in sequence. Hence the expected output is a string containing names separated by a colon of the 3 persons who hold these tokens. Note that the names are in ascending sequence of their respective token numbers

Example 2:

Input1= 17

Input2={“aa","bb",” cc”,” dd”, "ee”, “ff”,”gg")

Input3 = {9,89,5,0,6,65,4}

Expected output ="gg:cc:ee "

Explanation: Out of the 7 token numbers, the numbers 4, 5 and 6 are in sequence Hence the expected output is a string containing names separated by a colon of the 3 persons who hold these tokens. Note that the names are in ascending sequence of their respective token numbers

Example 3:

Input1 = 4

Input2 = {"Priya", "Soumya" ,"Sam”, “Vidya”}

Input3 = {9,76,8,23}

Expected output = "NONE"

Explanation: The given 4 token numbers do not contain any sequence of 3 numbers. Hence, the expected output is a string containing NONE

Code:

```
int i,x=0,y=0,z=0,flag=0;
int[] a=input3.clone();
Arrays.sort(a);
for(i=0;i<input1-2;i++)
 {
    if(a[i+1]-a[i]==1 && a[i+2]-a[i+1]==1)
    {
            flag=1;
            x=a[i];
            y=a[i+1];
            z=a[i+2];
            break;
     }
 }
 if(flag==0)
return "NONE";

int fi=0,si=0,ti=0;
for(i=0;i<input1;i++)
 {
    if(x==input3[i])
            fi=i;
    if(y==input3[i])
            si=i;
    if(z==input3[i])
            ti=i;
}
return input2[fi]+":"+input2[si]+":"+input2[ti];
```

29) USER ID GENERATION (Model-1)

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-d 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-should be generated as below –

Last Letter of the smaller name + Entre word of the longer name + Digit at position N in the PIN when traversing 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 = Ray

Last Name =Roy

PIN = 560037

N= 6

Step1 - Length of Last_Name is less than the Length of First_Name, so the Smaler Name is “Roy” and the Longer Name is “Rajiv”

Step2 - The user id will be = Last Letter of the smaller name +Entre 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”+ Entre 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 alphabet in the user-id. So,user-id = YrAJIV75

ANSWER:

```
class UserMainCode
{
public String userIdGeneration(String input1,String input2,int input3,int input4){
int s1=input1.length();
int s2=input2.length();
String longer="";
String smaller="";
String output1="";
if(s1==s2)
{
if(input1.compareTo(input2)>0)
{
```

```
longer=input1;
smaller=input2;
}
else
{
longer=input2;
smaller=input1;
}
}
if(s1>s2){
longer=input1;
smaller=input2;
}
else if(s1<s2)
{
longer=input2;
smaller=input1;
}
String pin=input3+"";
String output=smaller.charAt(smaller.length()-1)+longer+pin.charAt
(input4-1)+pin.charAt(pin.length()-input4);
for(int i=0;i<output.length();i++)
{
if(Character.isLowerCase(output.charAt(i)))
{
output1+=Character.toUpperCase
(output.charAt(i));
}
else
{
output1+=Character.toLowerCase
(output.charAt(i));
}
}
return output1;
}
}
```

30) USER ID GENERATION (Model-2)

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-d 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-should be generated as below –

First Letter of the Longer name + Entire word of the Smaller name + Digit at position N in the PIN when traversing 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

Step 1 - Length of Last_Name is less than the Length of First_Name so the Smaller Name is “Roy” and the Longer Name is “Rajiv”

Step 2 - The user-id will be= First Letter of the longer name + Entire word of the smaller 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

=First Letter of “Rajiv” +Entire word of “Roy” + 6th Digit PIN from left + 6th Digit of PIN from right

=R+Roy + 7+5

Therefore, user-id =RRoy75

Step 3 - Toggle the alphabets in the user-id, user-id= rrOY75

ANSWER:

```
import java.util.*;

public class MyClass {

   public String userIdGeneration(String input1,String input2,int input3,int input4)

{

String s="",small="",longer="";

if(input1.length()<input2.length()){
```

```
small=input1;
longer=input2;
}
else if(input1.length()>input2.length())
{
small=input2;
longer=input1;
}
else
{
if((input1.compareTo(input2))<0)
{
small=input1;
longer=input2;
}
else
{
small=input2;
longer=input1;
}
}
s+=String.valueOf(longer.charAt(0))+small;
String pin=String.valueOf(input3);
String pinrev=String.valueOf(new StringBuilder(pin).reverse());
s+=String.valueOf(pin.charAt(input4-1))+String.valueOf(pinrev.charAt(input4-1));
String s1="";
for(int i=0;i<s.length()-2;i++){
if(Character.isUpperCase(s.charAt(i)))
```

```
s1+=String.valueOf(Character.toLowerCase(s.charAt(i)));

else

s1+=String.valueOf(Character.toUpperCase(s.charAt(i)));

}

s1+=s.substring(s.length()-2,s.length());

return s1;

}

}
```

31.) **USER ID GENERATION (MODEL-3 WITH DIFFERENT TEST CASE)**

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 islonger 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 longer name + Entire word of the smaller 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

Step 3 - 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 participants 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 longer name + Entire word of the smaller 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 “Rajiv” + Entire word of “Roy” + 6^{th} Digit of pin from left + 6^{th} Digit of pin from right

= v + Roy + 7 + 5

Therefore, user-id = vRoy75

Step3 -Toggle the alphabets in the user-id = VrOY75

ANSWER

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

// Read only region start
class UserIDGeneration {

    public String userIdGeneration(String input1,String input2,int input3,int input4){
    // Read only region end
        String firstName = input1;
        String lastName = input2;
        int pin = input3;
        int N = input4;

        String longerName;
        String smallerName;
        StringBuilder userId = new StringBuilder();

        if (firstName.length() >lastName.length()) {
            longerName = firstName;
```

```
                    smallerName = lastName;
                } else if (firstName.length() <lastName.length())  {
                    longerName = lastName;
                    smallerName = firstName;
                }
else
{
if (firstName.compareTo(lastName) <1 ) {
                    longerName = lastName;
                    smallerName = firstName;
                    }
else
{
                    longerName = firstName;
                    smallerName = lastName;

}
}
userId.append(longerName.charAt(longerName.length() - 1));
userId.append(smallerName);
for (int i = 0; i<userId.length(); i++)
{
if (Character.isUpperCase(userId.charAt(i)))
userId.setCharAt(i, Character.toLowerCase(userId.charAt(i)));
else
userId.setCharAt(i, Character.toUpperCase(userId.charAt(i)));
}
userId.append(String.valueOf(pin).charAt(N - 1));
userId.append(String.valueOf(pin).charAt(String.valueOf(pin).length() - N));
return userId.toString();
}
}
```

31) ENCODED THREE STRINGS (Model-1)

Anand was assigned the task of coming up with an encoding mechanism for any given three strings. He has come up with the following plan.

Step ONE: Given any three strings, break each string into 3 parts each.
For example- if the three strings are below:

Input 1: “John”
Input 2: “Johny”
Input 3: “Janardhan”
“John” should be split into “J”, “oh”, “n,” as the FRONT, MIDDLE and END part, respectively.
“Johny” should be split into “Jo”, “h”, “ny” as the FRONT, MIDDLE and END, respectively.
“Janardhan” should be split into “Jan”, “ard”, “han” as the FRONT, MIDDLE and END part, respectively.

i.e.

1) 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 “Janadhan”.
2) 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”.
3) 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 string as per the below specified concatenation – rule to from three Output strings.

Output1: FRONT part of input 1 + FRONT part of input 2 + FRONT part of input 3
Output2: MIDDLE part of input1 + MIDDLE part of input2 + MIDDLE part of input3
Output3: END part of the input1 + END part of input2 + END part of input3

For example, for the above example input strings:
Output1 = “J” + “Jo” + “Jan” = “JJoJan”
Output2 = “oh” + “h” + “ard” = “ohhard”
Output3 = “n” + “ny” + “han” = “nnyhan”

Step THREE:
Process the resulting output strings based on the output-processing rule. After the above two steps, we will now 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 i.e. for the above example.

Output1 = “JJoJan”
Output2= “ohhard’
Output3 = “NNYHAN”
Help Anand to write a program that would do the above.

Code:

```
String[][] res = new String[3][3];
String[] words = {input1, input2, input3};
for(int i=0;i<3;i++)
{
    int l = words[i].length();
    if(l%3==0 || l%3==1)
     {
            res[i][0] = words[i].substring(0, l/3);
            res[i][1] = words[i].substring(l/3, l-l/3);
            res[i][2] = words[i].substring(l-l/3);
    }
    else
    {
            res[i][0] = words[i].substring(0, l/3+1);
            res[i][1] = words[i].substring(l/3+1, l-l/3-1);
            res[i][2] = words[i].substring(l-l/3-1);
    }
}
//Output1: FRONT part of input 1 + FRONT part of input 2 + FRONT part of input 3
String output1 = res[0][0] + res[1][0] + res[2][0];

//Output2: MIDDLE part of input1 + MIDDLE part of input2 + MIDDLE part of input3
String output2 = res[0][1] + res[1][1] + res[2][1];

//Output3: END part of the input1 + END part of input2 + END part of input3
String output3 = res[0][2] + res[1][2] + res[2][2];

String temp=output3;
output3="";
for(int i=0;i<temp.length();i++)
{
    if(Character.isUpperCase(temp.charAt(i)))
            output3 = output3 + Character.toLowerCase(temp.charAt(i));
    else
            output3 = output3 + Character.toUpperCase(temp.charAt(i));
}
return new Result(output1, output2, output3);
```

32) ENCODED THREE STRINGS (Model-2)

Anand was assigned the task of coming up with an encoding mechanism for any given three strings. He has come up with the following plan.

Step ONE: Given any three strings, break each string into 3 parts each.
For example- if the three strings are below:
Input 1: “John”
Input 2: “Johny”
Input 3: “Janardhan”
“John” should be split into “J”, “oh”, “n,” as the FRONT, MIDDLE and END part, respectively.
“Johny” should be split into “Jo”, “h”, “ny” as the FRONT, MIDDLE and END, respectively.
“Janardhan” should be split into “Jan”, “ard”, “han” as the FRONT, MIDDLE and END part, respectively.

i.e.

1) 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 “Janadhan”.
2) 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”.
3) 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 string as per the below specified concatenation – rule to from three Output strings.
Output1: FRONT part of input 1 + MIDDLE part of input 2 + END part of input 3
Output2: MIDDLE part of input1 + END part of input2 + FRONT part of input3
Output3: END part of the input1 + FRONT part of input2 + MIDDLE part of input3

For example, for the above example input strings:
Output1 = “J” + “h” + “han” = “Jhhan”
Output2 = “oh” + “ny” + “Jan” = “ohnyJan”
Output3= “n” + “Jo” + “ard” += “nJoard”

Step THREE:
Process the resulting output strings based on the output-processing rule. After the above two steps, we will now 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 “nJoard”, so after applying the toggle rule. Output3 should become “NjOARD”.
Final Result – The three output strings after applying the above three steps i.e. for the above example.
Output1 = “Jnhan”
Output2= “ohnyJan’

Output3 = “NjOARD”
Help Anand to write a program that would do the above.

Code:

```
String[][] res = new String[3][3];
String[] words = {input1, input2, input3};
for(int i=0;i<3;i++)
{
    int l = words[i].length();
    if(l%3==0 || l%3==1)
     {
            res[i][0] = words[i].substring(0, l/3);
            res[i][1] = words[i].substring(l/3, l-l/3);
            res[i][2] = words[i].substring(l-l/3);
    }
    else
    {
            res[i][0] = words[i].substring(0, l/3+1);
            res[i][1] = words[i].substring(l/3+1, l-l/3-1);
            res[i][2] = words[i].substring(l-l/3-1);
    }
}
//Output1: FRONT part of input 1 + MIDDLE part of input 2 + END part of input 3
String output1 = res[0][0] + res[1][1] + res[2][2];

//Output2: MIDDLE part of input1 + END part of input2 + FRONT part of input3
String output2 = res[0][1] + res[1][2] + res[2][0];

//Output3: END part of the input1 + FRONT part of input2 + MIDDLE part of input3
String output3 = res[0][2] + res[1][0] + res[2][1];

String temp=output3;
output3="";
for(int i=0;i<temp.length();i++)
{
    if(Character.isUpperCase(temp.charAt(i)))
            output3 = output3 + Character.toLowerCase(temp.charAt(i));
    else
            output3 = output3 + Character.toUpperCase(temp.charAt(i));
}
return new Result(output1, output2, output3);
```

33) ENCODED THREE STRINGS (Model-3)

Anand was assigned the task of coming up with an encoding mechanism for any given three strings. He has come up with the following plan.

Step ONE: Given any three strings, break each string into 3 parts each.
For example- if the three strings are below:
Input 1: “John”
Input 2: “Johny”
Input 3: “Janardhan”
“John” should be split into “J”, “oh”, “n,” as the FRONT, MIDDLE and END part, respectively.
“Johny” should be split into “Jo”, “h”, “ny” as the FRONT, MIDDLE and END, respectively.
“Janardhan” should be split into “Jan”, “ard”, “han” as the FRONT, MIDDLE and END part, respectively.

i.e.

1) 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 “Janadhan”.
2) 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”.
3) 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 string as per the below specified concatenation – rule to from three Output strings.
Output1: FRONT part of input 1 + END part of input 2 + MIDDLE part of input 3
Output2: MIDDLE part of input1 + FRONT part of input2 + END part of input3
Output3: END part of the input1 + MIDDLE part of input2 + FRONT part of input3

For example, for the above example input strings:
Output1 = “J” + “nh” + “ard” = “Jnyard”
Output2 = “oh” + “Jo” + “han” = “ohJohan”
Output3= “n” + “h” + “Jan” += “nhJan”

Step THREE:
Process the resulting output strings based on the output-processing rule. After the above two steps, we will now 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 “nhJan”, so after applying the toggle rule. Output3 should become “NHjAN”.
Final Result – The three output strings after applying the above three steps i.e. for the above example.

Output1 = “Jnyard”
Output2= “ohJohan’
Output3 = “NHjAN”
Help Anand to write a program that would do the above.

Code:

```
String[][] res = new String[3][3];
String[] words = {input1, input2, input3};
for(int i=0;i<3;i++)
{
    int l = words[i].length();
    if(l%3==0 || l%3==1)
     {
            res[i][0] = words[i].substring(0, l/3);
            res[i][1] = words[i].substring(l/3, l-l/3);
            res[i][2] = words[i].substring(l-l/3);
    }
    else
    {
            res[i][0] = words[i].substring(0, l/3+1);
            res[i][1] = words[i].substring(l/3+1, l-l/3-1);
            res[i][2] = words[i].substring(l-l/3-1);
    }
}
// Output1: FRONT part of input 1 + END part of input 2 + MIDDLE part of input 3
String output1 = res[0][0] + res[1][2] + res[2][1];

// Output2: MIDDLE part of input1 + FRONT part of input2 + END part of input3
String output2 = res[0][1] + res[1][0] + res[2][2];

// Output3: END part of the input1 + MIDDLE part of input2 + FRONT part of input3
String output3 = res[0][2] + res[1][1] + res[2][0];

String temp=output3;
output3="";
for(int i=0;i<temp.length();i++)
{
    if(Character.isUpperCase(temp.charAt(i)))
            output3 = output3 + Character.toLowerCase(temp.charAt(i));
    else
            output3 = output3 + Character.toUpperCase(temp.charAt(i));
}
return new Result(output1, output2, output3);
```

34) ENCODED THREE STRINGS (Model-4)

Anand was assigned the task of coming up with an encoding mechanism for any given three strings. He has come up with the following plan.

Step ONE: Given any three strings, break each string into 3 parts each.
For example- if the three strings are below:
Input 1: “John”
Input 2: “Johny”
Input 3: “Janardhan”
“John” should be split into “J”, “oh”, “n,” as the FRONT, MIDDLE and END part, respectively.
“Johny” should be split into “J”, “ohn”, “y” as the FRONT, MIDDLE and END, respectively.
“Janardhan” should be split into “Jan”, “ard”, “han” as the FRONT, MIDDLE and END part, respectively.

i.e.
1) If the length of the given word can be equally divided into three parts, then each part gets the same number of letters (as seen in above examples of "John" and "Johny")
2) If the length of the given word cannot be equally divided into three parts, then the center part i.e., part2 gets the extra number of characters (as seen in “Janardhan”)

Step TWO: Concatenate (join) the FRONT, MIDDLE and END parts of the string as per the below specified concatenation – rule to from three Output strings.
Output1: FRONT part of input 1 + MIDDLE part of input 2 + END part of input 3
Output2: MIDDLE part of input1 + END part of input2 + FRONT part of input3
Output3: END part of the input1 + FRONT part of input2 + MIDDLE part of input3

For example, for the above example input strings:
Output1 = “J” + “ohn” + “han” = “Johnhan”
Output2 = “oh” + “y” + “Jan” = “ohyJan”
Output3= “n” + “J” + “ard” += “nJard”

Step THREE:
Process the resulting output strings based on the output-processing rule. After the above two steps, we will now have three output string. 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 “nJard”, so after applying the toggle rule. Output3 should become “NjARD”.

Final Result – The three output strings after applying the above three steps i.e. for the above example.
Output1 = “Johnhan”
Output2= “ohyJan’
Output3 = “NjARD”

Help Anand to write a program that would do the above.

Code:

```
String[][] res = new String[3][3];
String[] words = {input1, input2, input3};
for(int i=0;i<3;i++)
{
    int l = words[i].length();
    res[i][0] = words[i].substring(0, l/3);
    res[i][1] = words[i].substring(l/3, l-l/3);
    res[i][2] = words[i].substring(l-l/3);
}
//Output1: FRONT part of input 1 + MIDDLE part of input 2 + END part of input 3
String output1 = res[0][0] + res[1][1] + res[2][2];

//Output2: MIDDLE part of input1 + END part of input2 + FRONT part of input3
String output2 = res[0][1] + res[1][2] + res[2][0];

//Output3: END part of the input1 + FRONT part of input2 + MIDDLE part of input3
String output3 = res[0][2] + res[1][0] + res[2][1];

String temp=output3;
output3="";
for(int i=0;i<temp.length();i++)
{
    if(Character.isUpperCase(temp.charAt(i)))
            output3 = output3 + Character.toLowerCase(temp.charAt(i));
    else
            output3 = output3 + Character.toUpperCase(temp.charAt(i));
}
return new Result(output1, output2, output3);
```

35) LARGEST POSSIBLE PALINDROME

Remove characters from a word form largest possible Palindrome:

What is a Palindrome?

Palindrome is a string that spells the same from either directions, for example abba, appa, amma, malayalam, nayan, deed, level, madam, rotator, reviver stats, tenet…

Mohan was taught about palindromes at school today and he got fascinated by the idea of palindromes. He started analyzing various words and thought it should be possible to create palindromes from most words by removing a few characters from the word. Write a method to help Mohan find the number of characters to be removed from a given word so that the remaining characters in the word can form palindrome

NOTE: You are not expected to form one or all possible palindromes in the word You are expected to only find the number of characters that have to be removed from the word so that the remaining characters can form a palindrome

For example, the given word is Template If ‘m’, ‘p’,’l’ and ‘a’ are removed we are left with “Tete” which is a good candidate to form a palindrome. In addition, if we let one of the characters,’m’,’p’,’l’ or ‘a’ stay within the word, we can still form valid palindromes. For e.g we remove ‘m’,’l’ and ‘a’ but not ‘p’, then the set of characters in the word Would be “Tepte” which when re-arranged can form palindromes Such as "Tepet” or “Etpte” So 3 is the number of characters that have to be removed from “Template” so that the remaining characters can form he largest possible palindrome

The given function accepts one parameter input1 representing the word mat needs to be analyzedThe method should return the number of characters to be removed

NOTE 1: if all the characters in the word are already sufficient to form a palindrome, then the number of characters that have to be removed from the word should be 0.

For e.g the word is "Magma", then the result should be 0

NOTE 2: if all the characters in the word are different and cannot form a palindrome, then the number of characters that have to be removed from the word should be -1

For e.g the word is “Victory” then the result should be -1

NOTE 3: Ignore the case of the letters white doing the check i.e "Template” or “template" or “TEMplate” or “TEmPLAte” or “TEMPLATE” should all give the same result which is 3

NOTE 4: You can assume that the given word will be a single word with no spaces and only alphabet characters.

ANSWER:

```
   public int palindrome(String input1){

      input1=input1.toLowerCase();

      int h[]=new int[26];

for(int i=0;i<input1.length();i++){

         h[(int)input1.charAt(i)-97]++;
```

```
        }
        int c=0;
for(int i=0;i<26;i++){
            if(h[i]%2==1)
c++;
        }
if(c==0 || c==1)
        return 0;
        else if(c==input1.length())
        return -1;
        else
        return c-1;
    }
```

36) WEIGHT OF HILL PATTERNS

Given,

the total levels(rows) in a hill pattern (input1),

the weight of the head level(first row) as input2, and

the weight increments of each subsequent row as input3,

you are expected to find the total weight of the hill pattern.

"Total levels" represents the number of rows in the pattern.

"Head level" represents the first row.

Weight of a level represents the value of each star (asterisk) in that row.

Note that the first row will have the weight of the head level,and the weight of each subsequent row will keep increasing by the specified "weight increment".

The hill patterns will always be of the below format, starting with 1 star at head level and increasing 1 star at each level till level N.From second level(second row) a hash # also gets added to the pattern.

*

#

##*

##*#*

##*#*#*

##*#*#*#*

. . .and so on till level N

While the weight of a star * is equal to the weight of the current level(current row),the weight of the hash # is equal to the weight of the previous level(previous row)

Let us see a couple of examples.

Example1 -

Given,

the total levels(total rows) in the hill pattern = 5 (input1)

the weight of the head level (first row) = 10(input2)

the weight increments of each subsequent level = 2(input3)

Then, The total weight of the hill pattern will be calculated as = 10 + (12+10+12) + (14+12+14+12+14) + (16+14+16+14+16+14+16) + (18+16+18+16+18+16+18+16+18) = 10 + 34 + 66 + 106 + 154 = 370

Example2 -

Given,

the total levels in the hill pattern = 4(input1)

the weight of the head level = 1(input2)

the weight increments of each subsequent level = 5(input3)

Then, Total weight of the hill pattern will be = 1 + (6+1+6) + (11+6+11+6+11) + (16+11+16+11+16+11+16) = 1 + 13 + 45 + 97 = 156

ANSWER:

```
public static void Hill(int input1,int input2,int input3) {
```

```
int hash;

hash=input2;

int sum1=0,sum=0;

for(int i=0;i<input1;i++)

{

for(int j=0;j<=i;j++)

{

sum=sum+input2;

}

input2=input2+input3;

for(int k=0;k<=i&&i!=input2-1;k++)

{

sum1=sum1+hash;

}

hash=hash+input3;

}

return sum;

    }

}
```

37) JUMBLED WORDS

Wipro plans to publish a daily online newsletter on its internal website channelW,and you have been contacted to contribute to the JUMBLE game on the newsletter's fun page.The JUMBLE game presents a sentence with each word jumbled,and the readers will be expected to unjumble them.

Your task is to provide the sentence in jumbled format, i.e each word in the sentence jumbled in some fashion.

Because your task is to provide a jumbled sentence every day,you decide to write a program that can take any proper sentence as input, and jumble the words in that sentence to produce a jumbled sentence.

After some thought you decide below two ways of jumbling a word.....

Method-1(forward,backward):In the word,reading left to right(forward),Pick every odd letter starting from the first letter,and then reading the word from right to left(backward),pick every even letter starting from the last even letter in the word.

For example,

If the word is "PROJECT"the jumbled word will be "POETCJR"

Similarly,

If the word is "LEARNING" the jumbled word will be "LANNGIRE"

So,If the sentence is "PROJECT BASED LEARNING",the sentence with jumbled words should be "POETCJR BSDEA LANNGIRE".

Method-2(forward,forward):In the word,reading from left to right(forward),Pick every odd letter starting from the first letter, and then reading the word again from left to right(forward),pick every even letter starting from the second letter in the word

For example,

If the word is "PROJECT", the jumbled word will be "POETRJC".

Similarly,

If the word is "LEARNING" the jumbled word will be "LANNERIG"

So,If the sentence is "PROJECT BASED LEARNING",the sentence with jumbled words should be "POETRJC BSDAE LANNERIG".

The function JumbledWords takes two inputs-

input1 which reperesents the string(proper sentence)containing one or more words that are to be jumbled.

input2 which represents the type of jumbling method(1 0r 2)

The function is expected to jumble the words in given sentence(input1) based on the jumbling method specified(input2) and return the result(i.e sentence with each word jumbled).

Example1:

input1="PROJECT BASED LEARNING"

input2=1

Expected output="POETCJR BSDEA LANNGIRE"

Example2:

input1="PROJECT BASED LEARNING"

input2=2

Expected output="POETRJC BSDAE LANNERIG"

Example3:

input1="WIPRO LIMITED"

input2=1

Expected output="WPORI LMTDEII"

Example4:

input1="WIPRO LIMITED"

input2=2

Expected output="WPOIR LMTDIIE"

ANSWER:

```
   public String jumbledWords(String input1,int input2){
      String s[]=input1.split(" ");
      String s1="",res="",even="",odd="";
for(int i=0;i<s.length;i++){
         s1=s[i];
         even="";
         odd="";
for(int j=0;j<s1.length();j++){
            if(j%2==0)
            even+=String.valueOf(s1.charAt(j));
            else
```

```
            odd+=String.valueOf(s1.charAt(j));
        }
        if(input2==1)
        res+=even+String.valueOf(new StringBuffer(odd).reverse())+" ";
        else
        res+=even+odd+" ";
    }
    return res;
  }
```

38) FIX THE FORMULA

Kely has been tricked by her brother to answer a question with a number. She is perplexed. Here is the question “Fo+23the3*like2+” it took time for her to understand. Now she wants to automate it with a program so that any time her brother comes with such tricky String she could answer with lesser efforts.

Here is what we have to do, separate the math operators and the digits.

Like in the above String you can see the operators (+,*,+) and digits (2,3,3,2).

Rest all characters are ignored

No arrange the digits and operators in the order of the appearance to get the correct result.

2+3*3+2 to be solved as

(2+3) = 5

Then, (5*3)=15

Then (15+2) = 17

So for the given String **Fo+23the3*like2+**final answer is 17

Help kely by writing a program to solve the above given problem.

Prototype: Public int fix TheFormula(String inpt1)

Assumptions:

1. Numbers present in the String are always considered as single digits(0-9)
2. Only operators used in the String are(+,-,*,/)
3. Always we will have length +1 numbers to operators (int the above example 3 operators and 4 numbers)

Sample Input/Output-1

Input1= we8+you2-7to/*32

Output=2

Explanation: Here the operators are [+,-,/,*] and the numbers are [8,2,7,3,2]

Thus we would be getting 8+2=>10-7=>3/3=>1*2=>2

Final answer is 2.

Sample Input/Output-2

Input1= i*-t5s-t8h1e4birds

Output=35

Explanation: Here the operators are [+,-,-] and the numbers are [5,8,1,4]

Thus we would be getting 5*8=>40-1=>39-4=>35

Final answer is 35.

ANSWER:

```
   public int fixTheFormula(String input1){
      int d[]=new int[input1.length()];
      char c[]=new char[input1.length()];
      int k1=0,k2=0;
for(int i=0;i<input1.length();i++){
         if(!Character.isLetter(input1.charAt(i))){
if(Character.isDigit(input1.charAt(i))){
               d[k1++]=Integer.parseInt(String.valueOf(input1.charAt(i)));
            }
else{
               c[k2++]=input1.charAt(i);
            }
         }
      }
      int res=d[0],k=1;
```

```
for(int i=0;i<k2;i++){
        if(c[i]=='+'){
           res+=d[k];
        }
        else if(c[i]=='-'){
           res-=d[k];
        }
        else if(c[i]=='*'){
           res*=d[k];
        }
else{
           res/=d[k];
        }
        k++;
     }
     return res;
  }
```

39) FORM THE WORD

Given a string input1, which contains many numbers of words separated by : and each word contains exactly two lower case alphabets, generate and output based upon the below 2 cases.

Note:

1. All the characters in input1 are lower case alphabets
2. Input1 will always contain more than one word separated by :
3. Output should be returned in UpperCase

Case 1:

Check whether the two alphabets are same

If yes then take one alphabet from it and add it to output

Example-1

Input1 = ww:ii:pp:rr:oo

Output= WIPRO

Explanation

Word1 is ww, both are same hence take w

Word2 is ii, both are same hence take i

Word3 is pp, both are same hence take p

Word4 is rr, both are same hence take r

Word5 is oo, both are same hence take o

Hence the output is WIPRO

Case 2:

If the two alphabets are not same, then find the position value of them and find maximum value – minimum value.

Take the alphabet which comes at this (maximum value – minimum value) position in the alphabet series.

Example-2

Input1= zx:za:ee

Output=BYE

Explanation

Word1 is zx, both are not same alphabets

Position value of z is 26

Position value of x is 24

Max-min will be 26-24 = 2

Alphabet which comes in 2nd position is b

Word2 is za, both are not same alphabets

Position value of z is 26

Position value of a is 1

Max-min will be 26-1 = 25

Alphabet which comes in 25th position is y

Word3 is ee, both are same hence take e

Hence the output is BYE

ANSWER:

```
import java.util.*;
public class MyClass {
   public String formTheWord(String input1){
      String s[]=input1.split(":");
      String s1="",res="";
      int x;
for(int i=0;i<s.length;i++){
         s1=s[i];
         if(((int)s1.charAt(0))-((int)s1.charAt(1))==0){
            res+=String.valueOf(s1.charAt(0));
         }
         else if(((int)s1.charAt(0))-((int)s1.charAt(1))>0){
            x=((int)s1.charAt(0))-((int)s1.charAt(1));
            res+=String.valueOf((char)(96+x));
         }
         else{
            x=((int)s1.charAt(1))-((int)s1.charAt(0));
            res+=String.valueOf((char)(96+x));
         }
      }
      return res.toUpperCase();
   }
   public static void main(String args[]) {
      Scanner sc=new Scanner(System.in);
```

```
        String s=sc.next();

System.out.println(new MyClass().formTheWord(s));

    }

}
```

40) TWO DIGIT REDUCED SUBTRACTED FORM

Given a number, you are expected to find its two-digit “Reduced Subtracted Form(RSF)”

The “Reduced Subtracted Form(RSF)” of a number can be found by concatenating the difference between its adjacent digits

To find the two-digit “Reduced Subtracted Form(RSF)”,we need to continue this process till the resultant RSF is not a two digit number.

For egif the input number is 6928, its RSF can be found by concatenating the difference between (6 and 9), (9 and 2) and (2 and 8) as shown below –

Difference between 6 and 9 is 3

Difference between 9 and 2 is 7

Difference between 2 and 8 is 6

So, the “Reduced Subtracted Form(RSF)” of 6928 = 376

The resultant RSF (376) is not a two-digit number, so we must continue finding its “Reduced Subtracted Form(RSF)”

Difference between 3 and 7 is 4

Difference between 7 and 6 is 1

So, the “Reduced Subtracted Form(RSF)” of 376 = 41

The resultant RSF (41) is two-digit number, so we have reached the “two-digit Reduced Subtracted Form”.

Therefore, the two-digit RSF of 6928 = 41

Let’s see another example

If input1 = 5271

Expected output = 21

Explanation:

RSF of 5271 = (5-2)(2-7)(7-1)=356

RSF of 356=(3-5)(5-6)=21

Note1: input1 will be always be >=100

Note2:Note that while concatenating the differences, we are expected to use the absolute values(non-negative)

Note3: The input values for all test cases in this program have been designed such that their two-digits RSF will definitely result in a two-digit number

ANSWER:

Two digit reduced subtracted form

```
while(input1>=100)
  {
   int x=input1,l=0;
   while(input1>0)
    {
     input1=input1/10;
     l++;
    }
   int a[]=new int[l];
   int i=l-1;
   while(x>0)
    {
     a[i]=x%10;
     x=x/10;
i--;
    }
for(int j=0;j<l-1;j++)
    {
     input1=input1*10+Math.abs(a[j]-a[j+1]);
    }
  }
  return input1;
```

41) MATCHING WORD

Given 2 String array s1 & s2 which contains x elements each. S1 contains jumbled words and s2 contains corresponding correct words but in same or different order.
Find the matching word from S2 with the jumbled word from S1. Store the index of S2 in the string S. Repeat the same for all the elements of S1 and concatenate the index values to the string S.
Prototype: String findMatchingWord(string[] input1, string[] input2, int input3)

Input:2 string array S1, S2 &1 integer representing total elements in array S1 or S2.
S1- Array of jumbled words
S2- Array of correct words with the same or different order.

Output: string **S** which holds concatenated index values of **S2.**

Note: Elements in both the arrays shall be in lower case

Example 1:

Input1: {arc, nep, tis}
Input2: {sit, car, pen}
Input3: 3
Output: 120

Explanation:

The jumbled word "**arc"** from**input1** match with correct word "**car"** from **input2.** Index of "**car"** is **1.** Similarly "**nep"** matches with "**pen**", index of "**pen**" is 2 and "**tis**" matches with "**sit**", whose index is 0. By concatenating all the index values we get the output as **120**.

Example 2:

Input1: {cnhul, estl, rakeb, ahev}
Input2: {lets, have, lunch, break}
Input3: 4
Output: 2031

Answer:

```
String res="";
for(int i=0;i<input3;i++)
    {
       char t1[] = input1[i].toCharArray();
       char t2[] = input2[i].toCharArray();

       //sorting

Arrays.sort(t1);
Arrays.sort(t2);


       input1[i]= new String(t1);
       input2[i]= new String(t2);
    }

for(int i=0;i<input3;i++)
    {
for(int j=0;j<input3;j++)
       {
          if(input1[i].equals(input2[j]))
          {
             res=res+j;
break;
```

```
        }
    }
}
return Integer.parseInt(res);
```

42) ROBO Movement (90 degrees, 1 step):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
    String c = input4.charAt(i)+"";
    if(c.equals("R"))
    {
        if(d.equals("N"))
            d="E";
        else if(d.equals("E"))
            d="S";
        else if(d.equals("S"))
            d="W";
        else if(d.equals("W"))
            d="N";
    }
    else if(c.equals("L"))
    {
        if(d.equals("N"))
            d="W";
        else if(d.equals("E"))
            d="N";
        else if(d.equals("S"))
            d="E";
        else if(d.equals("W"))
            d="S";
    }
```

```
    else
    {
        if(d.equals("N") && y+1<=input2)
            y=y+1;
        else if(d.equals("E") && x+1<=input1)
            x+=1;
        else if(d.equals("S") && y-1>=0)
            y-=1;
        else if(d.equals("W") && x-1>=0)
            x-=1;
        else
            return x+"-"+y+"-"+d+"-ER";
    }
}
return x+"-"+y+"-"+d;
```

43) ROBO Movement (90 degrees, 2 steps):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];

for(int i=0;i<input4.length();i+=2)
{
    String c = input4.charAt(i)+"";
    if(c.equals("R"))
    {
        if(d.equals("N"))
            d="E";
        else if(d.equals("E"))
            d="S";
        else if(d.equals("S"))
            d="W";
        else if(d.equals("W"))
            d="N";
    }
    else if(c.equals("L"))
    {
        if(d.equals("N"))
            d="W";
        else if(d.equals("E"))
            d="N";
        else if(d.equals("S"))
            d="E";
        else if(d.equals("W"))
            d="S";
```

```
    }
    else
    {
        if(d.equals("N") && y+2<=input2)
            y+=2;
        else if(d.equals("E") && x+2<=input1)
            x+=2;
        else if(d.equals("S") && y-2>=0)
            y-=2;
        else if(d.equals("W") && x-2>=0)
            x-=2;
        else
            return x+"-"+y+"-"+d+"-ER";
    }
  }
  return x+"-"+y+"-"+d;
```

44) ROBO Movement (90 degrees, 1 or 2 steps):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
    String c = input4.charAt(i)+"";
    if(c.equals("R"))
    {
        if(d.equals("N"))
            d="E";
        else if(d.equals("E"))
            d="S";
        else if(d.equals("S"))
            d="W";
        else if(d.equals("W"))
            d="N";
    }
    else if(c.equals("L"))
    {
        if(d.equals("N"))
            d="W";
        else if(d.equals("E"))
            d="N";
        else if(d.equals("S"))
            d="E";
        else if(d.equals("W"))
            d="S";
    }
```

```
    else
    {
        int m= c.equals(“m”)?1:2;
        if(d.equals("N") && y+m<=input2)
            y+=m;
        else if(d.equals("E") && x+m<=input1)
            x+=m;
        else if(d.equals("S") && y-m>=0)
            y-=m;
        else if(d.equals("W") && x-m>=0)
            x-=m;
        else
            return x+"-"+y+"-"+d+"-ER";
    }
  }
  return x+"-"+y+"-"+d;
```

45) ROBO Movement (45 degrees, 1 step):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
   String c = input4.charAt(i)+"";
   if(c.equals("r"))
   {
      if(d.equals("N"))
         d="NE";
      else if(d.equals("E"))
         d="SE";
      else if(d.equals("S"))
         d="SW";
      else if(d.equals("W"))
         d="NW";
      else if(d.equals("NE"))
         d="E";
      else if(d.equals("SE"))
         d="S";
      else if(d.equals("SW"))
         d="W";
      else if(d.equals("NW"))
         d="N";
   }
   else if(c.equals("l"))
   {
      if(d.equals("N"))
```

```
        d="NW";
    else if(d.equals("E"))
        d="NE";
    else if(d.equals("S"))
        d="SE";
    else if(d.equals("W"))
        d="SW";
    else if(d.equals("NE"))
        d="N";
    else if(d.equals("SE"))
        d="E";
    else if(d.equals("SW"))
        d="S";
    else if(d.equals("NW"))
        d="W";
}
else
{
    if(d.equals("N") && y+1<=input2)
        y+=1;
    else if(d.equals("E") && x+1<=input1)
        x+=1;
    else if(d.equals("S") && y-1>=0)
        y-=1;
    else if(d.equals("W") && x-1>=0)
        x-=1;
    else if(d.equals("NE") && x+1<=input1 && y+1<=input2)
    {
        x+=1;
        y+=1;
    }
    else if(d.equals("SE") && x+1<=input1 && y-1>=0)
    {
        x+=1;
        y-=1;
    }
    else if(d.equals("SW") && x-1>=0 && y-1>=0)
    {
        x-=1;
        y-=1;
    }
    else if(d.equals("NW") && x-1>=0 && y+1<=input2)
    {
        x-=1;
        y+=1;
    }
    else
        return x+"-"+y+"-"+d+"-ER";
```

```
    }
}
return x+"-"+y+"-"+d;
```

46) ROBO Movement (45 degrees, 2 steps):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
   String c = input4.charAt(i)+"";
   else if(c.equals("r"))
   {
      if(d.equals("N"))
         d="NE";
      else if(d.equals("E"))
         d="SE";
      else if(d.equals("S"))
         d="SW";
      else if(d.equals("W"))
         d="NW";
      else if(d.equals("NE"))
         d="E";
      else if(d.equals("SE"))
         d="S";
      else if(d.equals("SW"))
         d="W";
      else if(d.equals("NW"))
         d="N";
   }
   else if(c.equals("l"))
   {
      if(d.equals("N"))
```

```
        d="NW";
    else if(d.equals("E"))
        d="NE";
    else if(d.equals("S"))
        d="SE";
    else if(d.equals("W"))
        d="SW";
    else if(d.equals("NE"))
        d="N";
    else if(d.equals("SE"))
        d="E";
    else if(d.equals("SW"))
        d="S";
    else if(d.equals("NW"))
        d="W";
}
else
{
    if(d.equals("N") && y+2<=input2)
        y+=2;
    else if(d.equals("E") && x+2<=input1)
        x+=2;
    else if(d.equals("S") && y-2>=0)
        y-=2;
    else if(d.equals("W") && x-2>=0)
        x-=2;
    else if(d.equals("NE") && x+2<=input1 && y+2<=input2)
    {
        x+=2;
        y+=2;
    }
    else if(d.equals("SE") && x+2<=input1 && y-2>=0)
    {
        x+=2;
        y-=2;
    }
    else if(d.equals("SW") && x-2>=0 && y-2>=0)
    {
        x-=2;
        y-=2;
    }
    else if(d.equals("NW") && x-2>=0 && y+2<=input2)
    {
        x-=2;
        y+=2;
    }
    else
        return x+"-"+y+"-"+d+"-ER";
```

```
    }
  }
  return x+"-"+y+"-"+d;
```

47) ROBO Movement (45 degrees, 1or 2 steps):

```
{
  String[] robo = input3.split("-");
  int x = Integer.parseInt(robo[0]);
  int y = Integer.parseInt(robo[1]);
  String d = robo[2];
  for(int i=0;i<input4.length();i+=2)
  {
    String c = input4.charAt(i)+"";
    else if(c.equals("r"))
    {
      if(d.equals("N"))
        d="NE";
      else if(d.equals("E"))
        d="SE";
      else if(d.equals("S"))
        d="SW";
      else if(d.equals("W"))
        d="NW";
      else if(d.equals("NE"))
        d="E";
      else if(d.equals("SE"))
        d="S";
      else if(d.equals("SW"))
        d="W";
      else if(d.equals("NW"))
        d="N";
    }
    else if(c.equals("l"))
    {
```

```
    if(d.equals("N"))
       d="NW";
    else if(d.equals("E"))
       d="NE";
    else if(d.equals("S"))
       d="SE";
    else if(d.equals("W"))
       d="SW";
    else if(d.equals("NE"))
       d="N";
    else
       if(d.equals("SE"))
       d="E";
    else if(d.equals("SW"))
       d="S";
    else if(d.equals("NW"))
       d="W";
}
else
{
    int m = c.equals(“m”)?1:2;
    if(d.equals("N") && y+m<=input2)
       y=y+m;
    else if(d.equals("E") && x+m<=input1)
       x+=m;
    else if(d.equals("S") && y-m>=0)
       y-=m;
    else if(d.equals("W") && x-m>=0)
       x-=m;
    else if(d.equals("NE") && x+m<=input1 && y+m<=input2)
    {
       x+=m;
       y+=m;
    }
    else if(d.equals("SE") && x+m<=input1 && y-m>=0)
    {
       x+=m;
       y-=m;
    }
    else if(d.equals("SW") && x-m>=0 && y-m>=0)
    {
       x-=m;
       y-=m;
    }
    else if(d.equals("NW") && x-m>=0 && y+m<=input2)
    {
       x-=m;
       y+=m;
```

```
        }
        else
            return x+"-"+y+"-"+d+"-ER";
    }
}
return x+"-"+y+"-"+d;
```

48) ROBO Movement (90 or 45 degrees, 1 step):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
    String c = input4.charAt(i)+"";
    if(c.equals("R"))
    {
        if(d.equals("N"))
            d="E";
        else if(d.equals("E"))
            d="S";
        else if(d.equals("S"))
            d="W";
        else if(d.equals("W"))
            d="N";
        else if(d.equals("NE"))
            d="SE";
        else if(d.equals("SE"))
            d="SW";
        else if(d.equals("SW"))
            d="NW";
        else if(d.equals("NW"))
            d="NE";
    }
    else if(c.equals("r"))
    {
```

```
    if(d.equals("N"))
        d="NE";
    else if(d.equals("E"))
        d="SE";
    else if(d.equals("S"))
        d="SW";
    else if(d.equals("W"))
        d="NW";
    else if(d.equals("NE"))
        d="E";
    else if(d.equals("SE"))
        d="S";
    else if(d.equals("SW"))
        d="W";
    else if(d.equals("NW"))
        d="N";
}
else if(c.equals("L"))
{
    if(d.equals("N"))
        d="W";
    else if(d.equals("E"))
        d="N";
    else if(d.equals("S"))
        d="E";
    else if(d.equals("W"))
        d="S";
    else if(d.equals("NE"))
        d="NW";
    else if(d.equals("SE"))
        d="NE";
    else if(d.equals("SW"))
        d="SE";
    else if(d.equals("NW"))
        d="SW";
}
else if(c.equals("I"))
{
    if(d.equals("N"))
        d="NW";
    else if(d.equals("E"))
        d="NE";
    else if(d.equals("S"))
        d="SE";
    else if(d.equals("W"))
        d="SW";
    else if(d.equals("NE"))
        d="N";
```

```
        else if(d.equals("SE"))
            d="E";
        else if(d.equals("SW"))
            d="S";
        else if(d.equals("NW"))
            d="W";
    }
    else
    {
        int m = 1;
        if(d.equals("N") && y+m<=input2)
            y=y+m;
        else if(d.equals("E") && x+m<=input1)
            x+=m;
        else if(d.equals("S") && y-m>=0)
            y-=m;
        else if(d.equals("W") && x-m>=0)
            x-=m;
        else if(d.equals("NE") && x+m<=input1 && y+m<=input2)
        {
            x+=m;
            y+=m;
        }
        else if(d.equals("SE") && x+m<=input1 && y-m>=0)
        {
            x+=m;
            y-=m;
        }
        else if(d.equals("SW") && x-m>=0 && y-m>=0)
        {
            x-=m;
            y-=m;
        }
        else if(d.equals("NW") && x-m>=0 && y+m<=input2)
        {
            x-=m;
            y+=m;
        }
        else
            return x+"-"+y+"-"+d+"-ER";
    }
}
return x+"-"+y+"-"+d;
```

49) ROBO Movement (90 or 45 degrees, 2 step):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
   String c = input4.charAt(i)+"";
   if(c.equals("R"))
   {
      if(d.equals("N"))
         d="E";
      else if(d.equals("E"))
         d="S";
      else if(d.equals("S"))
         d="W";
      else if(d.equals("W"))
         d="N";
      else if(d.equals("NE"))
         d="SE";
      else if(d.equals("SE"))
         d="SW";
      else if(d.equals("SW"))
         d="NW";
      else if(d.equals("NW"))
         d="NE";
   }
   else if(c.equals("r"))
   {
      if(d.equals("N"))
         d="NE";
      else if(d.equals("E"))
         d="SE";
      else if(d.equals("S"))
         d="SW";
      else if(d.equals("W"))
         d="NW";
      else if(d.equals("NE"))
         d="E";
      else if(d.equals("SE"))
         d="S";
      else if(d.equals("SW"))
         d="W";
      else if(d.equals("NW"))
         d="N";
   }
   else if(c.equals("L"))
   {
```

```
    if(d.equals("N"))
        d="W";
    else if(d.equals("E"))
        d="N";
    else if(d.equals("S"))
        d="E";
    else if(d.equals("W"))
        d="S";
    else if(d.equals("NE"))
        d="NW";
    else if(d.equals("SE"))
        d="NE";
    else if(d.equals("SW"))
        d="SE";
    else if(d.equals("NW"))
        d="SW";
}
else if(c.equals("I"))
{
    if(d.equals("N"))
        d="NW";
    else if(d.equals("E"))
        d="NE";
    else if(d.equals("S"))
        d="SE";
    else if(d.equals("W"))
        d="SW";
    else if(d.equals("NE"))
        d="N";
    else if(d.equals("SE"))
        d="E";
    else if(d.equals("SW"))
        d="S";
    else if(d.equals("NW"))
        d="W";
}
else
{
    int m = 2; // 2 steps
    if(d.equals("N") && y+m<=input2)
        y=y+m;
    else if(d.equals("E") && x+m<=input1)
        x+=m;
    else if(d.equals("S") && y-m>=0)
        y-=m;
    else if(d.equals("W") && x-m>=0)
        x-=m;
    else if(d.equals("NE") && x+m<=input1 && y+m<=input2)
```

```
{
   x+=m;
   y+=m;
}
else if(d.equals("SE") && x+m<=input1 && y-m>=0)
{
   x+=m;
   y-=m;
}
else if(d.equals("SW") && x-m>=0 && y-m>=0)
{
   x-=m;
   y-=m;
}
else if(d.equals("NW") && x-m>=0 && y+m<=input2)
```

```
        {
            x-=m;
            y+=m;
        }
        else
            return x+"-"+y+"-"+d+"-ER";
    }
}
return x+"-"+y+"-"+d;
```

50) ROBO Movement (90 or 45 degrees, 1 or 2 steps):

```
String[] robo = input3.split("-");
int x = Integer.parseInt(robo[0]);
int y = Integer.parseInt(robo[1]);
String d = robo[2];
for(int i=0;i<input4.length();i+=2)
{
    String c = input4.charAt(i)+"";
    if(c.equals("R"))
    {
        if(d.equals("N"))
            d="E";
        else if(d.equals("E"))
            d="S";
        else if(d.equals("S"))
            d="W";
        else if(d.equals("W"))
            d="N";
        else if(d.equals("NE"))
            d="SE";
        else if(d.equals("SE"))
            d="SW";
        else if(d.equals("SW"))
            d="NW";
        else if(d.equals("NW"))
            d="NE";
    }
    else if(c.equals("r"))
    {
        if(d.equals("N"))
            d="NE";
        else if(d.equals("E"))
            d="SE";
        else if(d.equals("S"))
```

```
            d="SW";
        else if(d.equals("W"))
            d="NW";
        else if(d.equals("NE"))
            d="E";
        else if(d.equals("SE"))
            d="S";
        else if(d.equals("SW"))
            d="W";
        else if(d.equals("NW"))
            d="N";
    }
    else if(c.equals("L"))
    {
        if(d.equals("N"))
            d="W";
        else if(d.equals("E"))
            d="N";
        else if(d.equals("S"))
            d="E";
        else if(d.equals("W"))
            d="S";
        else if(d.equals("NE"))
            d="NW";
        else if(d.equals("SE"))
            d="NE";
        else if(d.equals("SW"))
            d="SE";
        else if(d.equals("NW"))
            d="SW";
    }
    else if(c.equals("I"))
    {
        if(d.equals("N"))
            d="NW";
        else if(d.equals("E"))
            d="NE";
        else if(d.equals("S"))
            d="SE";
        else if(d.equals("W"))
            d="SW";
        else if(d.equals("NE"))
            d="N";
        else if(d.equals("SE"))
            d="E";
        else if(d.equals("SW"))
            d="S";
        else if(d.equals("NW"))
```

```
            d="W";
        }
        else
        {
            int m = c.equals("m")?1:2;
            if(d.equals("N") && y+m<=input2)
                y=y+m;
            else if(d.equals("E") && x+m<=input1)
                x+=m;
            else if(d.equals("S") && y-m>=0)
                y-=m;
            else if(d.equals("W") && x-m>=0)
                x-=m;
            else if(d.equals("NE") && x+m<=input1 && y+m<=input2)
            {
                x+=m;
                y+=m;
            }
            else if(d.equals("SE") && x+m<=input1 && y-m>=0)
            {
                x+=m;
                y-=m;
            }
            else if(d.equals("SW") && x-m>=0 && y-m>=0)
            {
                x-=m;
                y-=m;
            }
            else if(d.equals("NW") && x-m>=0 && y+m<=input2)
            {
                x-=m;
                y+=m;
            }
            else
                return x+"-"+y+"-"+d+"-ER";
        }
    }
    return x+"-"+y+"-"+d;
```

51) ONE DIGIT REDUCED SUBTRACTED FORM

```
while(input1>=10)
  {
   int x=input1,l=0;
   while(input1>0)
   {
    input1=input1/10;
    l++;
   }
   int a[]=new int[l];
   int i=l-1;
   while(x>0)
   {
    a[i]=x%10;
    x=x/10;
i--;
   }
for(int j=0;j<l-1;j++)
   {
    input1=input1*10+Math.abs(a[j]-a[j+1]);
   }
  }
  return input1;
```

52) PROCESS TWO WORDS

Input1: Today is a nice day

Input2: 41

Output: ince doTday

Code:

```
String i1=input1;

int i2=input2;

int second=i2%10;

i2=i2/10;

int first=i2;

String arr[]=i1.split(" ");

String first_w=arr[first-1];

String second_w=arr[second-1];

String first_w1=first_w.substring(first_w.length()/2,first_w.length());
```

```
String first_w2;
if(first_w.length()%2==0)
{
   first_w2=first_w.substring(0,first_w.length()/2);
}
else
{
   first_w2=first_w.substring(0,(first_w.length()/2)+1);
}
StringBuilder i1_w=new StringBuilder();
i1_w.append(first_w2);
i1_w=i1_w.reverse();
String sw1=second_w.substring(second_w.length()/2,second_w.length());
String sw2;
if(second_w.length()%2==0)
{
  sw2=second_w.substring(0,second_w.length()/2);
}
else
{
   sw2=second_w.substring(0,(second_w.length()/2)+1);
}
StringBuilder iw2=new StringBuilder();
iw2.append(sw2);
iw2=iw2.reverse();
String ans=i1_w+first_w1+" "+iw2+sw1;
return ans;
```

53) Find Password – Stable Unstable (MODEL-1)

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 realizes 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. i.e., thefrequency of each digit in the number is the same. For e.g. 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 isNOT the same, for e.g., 221, 4314, 101, 233, 58135, 101 are examples ofunstable numbers.

The password can be found as below:
i.e. **Password=sum of all stable 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, andinput5=898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the **password should be=sum of all stable numbers=2003**

Code:

```
int[] num = {input1, input2, input3, input4, input5};
int stable=0, i, j;
for(i=0;i<5;i++)
{
    int[] freq = new int[10];  //  frequencies of all the digits
    int temp=num[i];
    int maxf=0;
    while(temp!=0)
    {
        int r = temp%10;
        freq[r]++;
        temp/=10;
        if(freq[r]>maxf)
            maxf=freq[r];
```

```
    }
    for(j=0;j<10;j++)
    {
        if(freq[j]!=0 &&freq[j]!=maxf)
            break;
    }
    if(j==10)
        stable+=num[i];
}
return stable;
```

54) Find Password – Stable Unstable (MODEL-2)

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 realizes 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. i.e., thefrequency of each digit in the number is the same. For e.g. 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 isNOT the same, for e.g., 221, 4314, 101, 233, 58135, 101 are examples ofunstable numbers.

The password can be found as below:
i.e. **Password= (number of stable numbers*10) - number of 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, andinput5=898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the **password should be = (number of stable numbers*10) - number of unstable numbers = (3*10) - 2 = 28**

Code:

```
    int[] num = {input1, input2, input3, input4, input5};
int stable=0, unstable=0, i, j;
```

```
for(i=0;i<5;i++)
{
    int[] freq = new int[10];  //  frequencies of all the digits
    int temp=num[i];
    int maxf=0;
    while(temp!=0)
    {
        int r = temp%10;
        freq[r]++;
        temp/=10;
        if(freq[r]>maxf)
            maxf=freq[r];
    }
    for(j=0;j<10;j++)
    {
        if(freq[j]!=0 &&freq[j]!=maxf)
            break;
    }
    if(j==10)
        stable++;
    else
        unstable++;
}
return (stable*10 - unstable);
```

55) Find Password – Stable Unstable (MODEL-3)

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 realizes 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. i.e., thefrequency of each digit in the number is the same. For e.g. 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 isNOT the same, for e.g., 221, 4314, 101, 233, 58135, 101 are examples ofunstable numbers.

The password can be found as below:

i.e. **Password= (number of unstable numbers*10) - number of stable 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, andinput5=898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the **password should be = (number of unstable numbers*10) - number of stable numbers = (2*10) - 3 = 17**

Code:

```
int[] num = {input1, input2, input3, input4, input5};

int stable=0, unstable=0, i, j;

for(i=0;i<5;i++)

{

    int[] freq = new int[10];  //  frequencies of all the digits

    int temp=num[i];

    int maxf=0;

    while(temp!=0)

    {

        int r = temp%10;

        freq[r]++;

        temp/=10;

        if(freq[r]>maxf)

            maxf=freq[r];

    }

    for(j=0;j<10;j++)

    {

        if(freq[j]!=0 &&freq[j]!=maxf)
```

```
            break;
    }
    if(j==10)
        stable++;
    else
        unstable++;
}
return (unstable*10 - stable);
```

56) Find Password – Stable Unstable (MODEL-4)

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 realizes 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. i.e., thefrequency of each digit in the number is the same. For e.g. 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 isNOT the same, for e.g., 221, 4314, 101, 233, 58135, 101 are examples ofunstable numbers.

The password can be found as below:
i.e. **Password= Maximum of all unstable numbers + Minimum 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, andinput5=898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the **password should be = Maximum of all unstable numbers + Minimum of all unstable numbers = 898+ 122 = 1020**

Code:

```
int[] num = {input1, input2, input3, input4, input5};
int max_unstable=Integer.MIN_VALUE, min_unstable=Integer.MAX_VALUE, i, j;
for(i=0;i<5;i++)
{
    int[] freq = new int[10];
    int temp=num[i];
    int maxf=0;
    while(temp!=0)
    {
        int r = temp%10;
        freq[r]++;
        temp/=10;
        if(freq[r]>maxf)
            maxf=freq[r];
    }
    for(j=0;j<10;j++)
    {
        if(freq[j]!=0 && freq[j]!=maxf)
            break;
    }

    if(j!=10)
    {
        if(num[i]>max_unstable)
            max_unstable=num[i];
        if(num[i]<min_unstable)
            min_unstable=num[i];
    }
}
if(max_unstable==Integer.MIN_VALUE)
    max_unstable=0;
if(min_unstable==Integer.MAX_VALUE)
    min_unstable=0;

return (max_unstable+min_unstable);
```

57) Find Password – Stable Unstable (MODEL-5)

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 realizes 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. i.e., thefrequency of each digit in the number is the same. For e.g. 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 isNOT the same, for e.g., 221, 4314, 101, 233, 58135, 101 are examples ofunstable numbers.
The password can be found as below:
i.e. **Password= Maximum of all unstable numbers + Minimum of all stable 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, andinput5=898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the **password should be = Maximum of all unstable numbers + Minimum of allstable numbers = 898 + 12 = 910**

Code:

```
int[] num = {input1, input2, input3, input4, input5};
int max_unstable=Integer.MIN_VALUE, min_stable=Integer.MAX_VALUE, i, j;
for(i=0;i<5;i++)
{
    int[] freq = new int[10];  //  frequencies of all the digits
    int temp=num[i];
    int maxf=0;
    while(temp!=0)
    {
        int r = temp%10;
        freq[r]++;
        temp/=10;
        if(freq[r]>maxf)
            maxf=freq[r];
    }
    for(j=0;j<10;j++)
    {
```

```
                if(freq[j]!=0 && freq[j]!=maxf)
                        break;
        }
        if(j==10&&num[i]<min_stable)
                min_stable=num[i];
        else if(j!=10 && num[i]>max_unstable)
                max_unstable=num[i];
}
if(max_unstable==Integer.MIN_VALUE)
        max_unstable=0;
if(min_stable==Integer.MAX_VALUE)
        min_stable=0;

return (max_unstable + min_stable);
```

58) Find Password – Stable Unstable (MODEL-6)

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 realizes 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. i.e., thefrequency of each digit in the number is the same. For e.g. 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 isNOT the same, for e.g., 221, 4314, 101, 233, 58135, 101 are examples ofunstable numbers.

The password can be found as below:
i.e. **Password= Maximum of all unstable numbers – Minimum of all stable 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, andinput5=898,
stable numbers are 12, 1313 and 678
unstable numbers are 122 and 898
So, the **password should be = Maximum of all unstable numbers – Minimum of all stable numbers = 898 –12 = 886**

Code:

```
int[] num = {input1, input2, input3, input4, input5};
int max_unstable=Integer.MIN_VALUE, min_stable=Integer.MAX_VALUE, i, j;
for(i=0;i<5;i++)
{
    int[] freq = new int[10];
    int temp=num[i];
    int maxf=0;
    while(temp!=0)
    {
        int r = temp%10;
        freq[r]++;
        temp/=10;
        if(freq[r]>maxf)
            maxf=freq[r];
    }
    for(j=0;j<10;j++)
    {
        if(freq[j]!=0 &&freq[j]!=maxf)
            break;
    }
    if(j==10&&num[i]<min_stable)
        min_stable=num[i];
    else if(j!=10 && num[i]>max_unstable)
        max_unstable=num[i];
}
if(max_unstable==Integer.MIN_VALUE)
    max_unstable=0;
if(min_stable==Integer.MAX_VALUE)
    min_stable=0;

return (max_unstable - min_stable);
```

59) Nambiar Number

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 th 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+2=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 the 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 = (26971) = 26971

Note: Please note that the number of passes required to process the given number may vary depending upon the constitution of the mobile number Note: Also note that. O 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][8117] = 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

Code:

```
 public 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());
}
```

60) Generate series and find nth element:

```
int gap1 = (input2 - input1);

 int gap2 = (input3 - input2);

 int output = input1;


for (int i = 1; i < input4; i++) {

  if (i % 2 == 1)

    output += gap1;

  else

    output += gap2;


}
```

```
return output;
```