71) Digits

Write a program to read a non-negative integer n, that returns the count of the occurances of 7 as digit.

Include a class UserMainCode with a static method **countSeven** which accepts the integer value. The return type is integer which is the count value.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a integer.
Output consists of integer.
Refer sample output for formatting specifications.

Sample Input 1:
717

Sample Output 1:
2

Sample Input 2:
4534

Sample Output 2:
0

```
public class User {
        public static int countSeven (int n)
        {
                int count=0,r=0;
                while(n>0)
                {
                        r=n%10;
                        if(r==7)
                                count++;
                        n=n/10;
                }
                return count;
        }
}
```

72) String Processing - III

Write a program to read a string where all the lowercase 'x' chars have been moved to the end of the string.

Include a class UserMainCode with a static method **moveX** which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
xxhixx

Sample Output 1:
hixxxx

Sample Input 2:
XXxxtest

Sample Output 2:
XXtestxx

```
public class User {
      public static String getStringUsingNthCharacter (String s)
      {
            StringBuffer sb=new StringBuffer();
            StringBuffer sb1=new StringBuffer();
            for(int i=0;i<s.length();i++)
            {
                  if(s.charAt(i)=='x')
                  {
                  Sb1.append(s.charAt(i));
                  }
                  else
                  {
                        sb.append(s.charAt(i));
                  }
            }
            sb.append(sb1);

            return sb.toString();
      }

}
```

73) String Processing - IV

Write a program to read a string and also a number N. Form a new string starting with 1st character and with every Nth character of the given string. Ex - if N is 3, use chars 1, 3, 6, ... and so on to form the new String. Assume N>=1.

Include a class UserMainCode with a static method **getStringUsingNthCharacter** which accepts the string and the number n. The return type is the string as per the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string and integer.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
HelloWorld
2

Sample Output 1:
HelWrd

```
public class User {
      public static String getStringUsingNthCharacter (String s,int n)
      {
            StringBuffer sb=new StringBuffer();
            sb.append(s.charAt(0));
            for(int i=1;i<s.length();i=i+n)
                  sb.append(s.charAt(i));
            return sb.toString();
      }

}
```

74) Digit Comparison

Write a program to read two integers and return true if they have the same last digit.

Include a class UserMainCode with a static method **compareLastDigit** which accepts two integers and returns boolean. (true / false)

Create a Class Main which would be used to accept two integers and call the static method present in UserMainCode.

Input and Output Format:

Input consists of two integer.
Output consists TRUE / FALSE.
Refer sample output for formatting specifications.

Sample Input 1:
59
29

Sample Output 1:
TRUE

```
public class User {
      public static boolean compareLastDigit   (int a,int b)
      {
            boolean b1=false;
            int r1=a%10;
            int r2=b%10;
            if(r1==r2)
                  b1=true;
            return b1;
      }
}
```

75) Duplicates

GIven three integers (a,b,c) find the sum. However, if one of the values is the same as another, both the numbers do not count towards the sum and the third number is returned as the sum.

Include a class UserMainCode with a static method **getDistinctSum** which accepts three integers and returns integer.

Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.

Input and Output Format:

Input consists of three integers.
Output consists of a integer.
Refer sample output for formatting specifications.

Sample Input 1:
1
2
1

Sample Output 1:
2

Sample Input 2:
1
2
3

Sample Output 2:
6

```
public class User {
        public static int getDistinctSum  (int a,int b,int c)
        {
                int sum=0;
                if(a==b&&a==c&&b==c)
                        sum=0;
                else if(a!=b&&b!=c&a==c)
                        sum=b;
                else if(a==b&&b!=c&&a!=c)
                        sum=c;
                else if(a!=b&&b!=c&&a!=c)
                        sum=a+b+c;
                                return sum;
        }
}
```

```
                                int sum=0;
                                if(a==b&&a==c&&b==c)
                                        sum=0;
                                else if(a!=b&&a!=c&b==c)
                                        sum=a;
                                else if(a!=b&&b!=c&a==c)
                                        sum=b;
                                else if(a==b&&b!=c&&a!=c)
                                        sum=c;
                                else
```

```
                              sum=a+b+c;
                                    return sum;
```

```
public class Main {
      public static void main(String[] args) {
                  Scanner sc=new Scanner(System.in);
            int a=sc.nextInt();
            int b=sc.nextInt();
            int c=sc.nextInt();
int sum=0;
            if(a!=b&&a!=c&&b!=c)
            {
                  sum=a+b+c;
            }

            else if(a==c)

            {
                  sum=b;
            }
            else if(a==b)
            {
                  sum=c;
            }
            else if(b==c)
            {
                  sum=a;
            }
            else
                  sum=0;
            System.out.println(sum);
      }
}
```

76) String Processing - MixMania

Write a program to read a string and check if it starts with '_ix' where '_' is any one char(a-z, A-Z, 0-9).

If specified pattern is found return true else false.

Include a class UserMainCode with a static method **checkPattern** which accepts the string. The return type is TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.
Output consists of TRUE / FALSE.
Refer sample output for formatting specifications.

Sample Input 1:
Mix Mania

Sample Output 1:
TRUE

```
public class User {
      public static boolean validateString (String s)
      {
            boolean b=false;
            if(s.charAt(1)=='i'&&s.charAt(2)=='x')
                  b=true;
            return b;
      }
}
```

77) String Processing

Write a program to read a string and return a new string where the first and last chars have been interchanged.

Include a class UserMainCode with a static method **exchangeCharacters** which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.
Output consists of string.
Refer sample output for formatting specifications.

Sample Input 1:
HelloWorld

Sample Output 1:

delloWorlH

```
public class HelloWorld {

    public static void main(String[] args) {
        // String s1="hello world";
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        String ss=Hello.display(s1);
        System.out.println(ss);

        }

    }

public class Hello {

    public static String display(String s1) {

        StringTokenizer st=new StringTokenizer(s1," ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String a=st.nextToken();
            String b=st.nextToken();
            sb.append(b.substring(b.length()-1));
            sb.append(a.substring(1));
            sb.append(" ");
            sb.append(b.substring(0,b.length()-1));
            sb.append(a.substring(0,1));
        }
        return sb.toString();
    }

}
```

```
public class WhiteSpaxc {
            public static String validateNumber(String s)
        {
                StringBuffer sb=new StringBuffer();
                    sb.append(s.substring(s.length()-1));
                    sb.append(s.substring(1,s.length()-1));
                    sb.append(s.substring(0,1));

                return sb.toString();
        }
    }
```

78) Regular Expression - II

Given a string (s) apply the following rules.

1. String consists of three characters only.
2. The characters should be alphabets only.

If all the conditions are satisifed then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validateString** which accepts the string. The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.
Output consists of TRUE or FALSE .
Refer sample output for formatting specifications.

Sample Input 1:
AcB

Sample Output 1:
TRUE

Sample Input 2:
A2B

Sample Output 2:
FALSE

```java
public class Main {
      public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            String s=sc.next();
            boolean b=User.validateString(s);
            System.out.println(b);
      }
}
```

```
public class User {
      public static boolean validateString (String s)
      {
            boolean b=false;
            if(s.length()==3)
            {
            if(s.matches("[A-za-z]{3}"))
                  b=true;
            }
            return b;
      }
}
```

79) Strings Processing - Replication

Write a program to read a string and also a number N. Return the replica of original string for n given time.

Include a class UserMainCode with a static method **repeatString** which accepts the the string and the number n. The return type is the string based on the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string and integer.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
Lily
2

Sample Output 1:
LilyLily

```
public class User {
      public static String repeatString(String s,int n)
      {
            StringBuffer sb=new StringBuffer();
   for(int i=0;i<n;i++)
   {
         sb.append(s);
   }
   return sb.toString();
      }
}
```

80) SumOdd

Write a program to read an integer and find the sum of all odd numbers from 1 to the given number. [inclusive of the given number]

if N = 9 [1,3,5,7,9]. Sum = 25

Include a class UserMainCode with a static method **addOddNumbers** which accepts the number n. The return type is the integer based on the problem statement.

Create a Class Main which would be used to accept the integer and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a integer.
Output consists of a integer.
Refer sample output for formatting specifications.

Sample Input 1:
6

Sample Output 1:
9

```
public class User {
      public static int SumOdd (int n)
      {
            int sum=0;
   for(int i=1;i<=n;i++)
   {
         if(i%2!=0)
               sum=sum+i;
   }
   return sum;
      }
}
```