61) Boundary Average

Given an int array as input, write a program to compute the average of the maximum and minimum element in the array.

Include a class **UserMainCode** with a static method "**getBoundaryAverage**" that accepts an integer array as argument and returns a float that corresponds to the average of the maximum and minimum element in the array.

Create a class **Main** which would get the input array and call the static method **getBoundaryAverage** present in the UserMainCode.

Input and Output Format:
The first line of the input consists of an integer n, that corresponds to the size of the array.
The next n lines consist of integers that correspond to the elements in the array.
Assume that the maximum number of elements in the array is 10.

Output consists of a single float value that corresponds to the average of the max and min element in the array.

Sample Input :
6
3
6
9
4
2
5

Sample Output:
5.5

```
public class User {
      public static float getBoundaryAverage(int a[],int n)
      {
         int sum=0;
         float avg=0;
         Arrays.sort(a);
         sum=a[0]+a[n-1];
         avg=(float)sum/2;
            return avg;
      }
}
```

62) Count Vowels

Given a string input, write a program to find the total number of vowels in the given string.

Include a class **UserMainCode** with a static method "**countVowels**" that accepts a String argument and returns an int that corresponds to the total number of vowels in the given string.

Create a class **Main** which would get the String as input and call the static method **countVowels** present in the UserMainCode.

Input and Output Format:
Input consists of a string.
Output consists of an integer..

Sample Input:
avinash

Sample Output:
3

```
public class User {
      public static int countVowels(String s) throws ParseException
      {
            int count=0;
        for(int i=0;i<s.length();i++)
        {
              if(s.charAt(i)=='a' ||s.charAt(i)=='A' ||
                          s.charAt(i)=='e' ||s.charAt(i)=='E' ||
                                s.charAt(i)=='i' ||s.charAt(i)=='I' ||
                                      s.charAt(i)=='o' ||s.charAt(i)=='O' ||
                                            s.charAt(i)=='u'
||s.charAt(i)=='U')
                        count++;
        }
        return count;

      }
}
```

63) Month Name

Given a date as a string input in the format dd-mm-yy, write a program to extract the month and to print the month name in upper case.

Include a class **UserMainCode** with a static method "**getMonthName**" that accepts a String argument and returns a String that corresponds to the month name.

Create a class **Main** which would get the String as input and call the static method **getMonthName** present in the UserMainCode.

The month names are {JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER}

Input and Output Format:
Input consists of a String.
Output consists of a String.

Sample Input:
01-06-82

Sample Output:
JUNE

```
public class User {
	public static String getMonthName(String s) throws ParseException
	{
		SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
		Date d=sdf.parse(s);

		SimpleDateFormat sdf1=new SimpleDateFormat("MMMM");
		String month=sdf1.format(d);

		return month.toUpperCase();

	}
}
```

64) Reverse SubString

Given a string, startIndex and length, write a program to extract the substring from right to left. Assume the last character has index 0.

Include a class **UserMainCode** with a static method "`public class User {`" that accepts 3 arguments and returns a string. The 1st argument corresponds to the string, the second argument corresponds to the startIndex and the third argument corresponds to the length.

Create a class **Main** which would get a String and 2 integers as input and call the static method **reverseSubstring** present in the UserMainCode.

Input and Output Format:
The first line of the input consists of a string.
The second line of the input consists of an integer that corresponds to the startIndex.

The third line of the input consists of an integer that corresponds to the length of the substring.

Sample Input:
rajasthan
2
3

Sample Output:
hts

```
import java.util.Scanner;
public class Main {
	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int n1=sc.nextInt();
int n2=sc.nextInt();
System.out.println(UserMainCode.reverseSubstring(s,n1,n2));

	}
}

public class UserMainCode {
	public static String reverseSubstring(String s,int n1,int n2)

	{
		StringBuffer sb=new StringBuffer(s);
		sb.reverse();
		String ss=sb.substring(n1,n1+n2);
		return ss.toString();
	}
}
```

65) String Finder

Given three strings say Searchstring, Str1 and Str2 as input, write a program to find out if Str2 comes after Str1 in the Searchstring.

Include a class **UserMainCode** with a static method "**stringFinder**" that accepts 3 String arguments and returns an integer. The 3 arguments correspond to SearchString, Str1 and Str2. The function returns 1 if Str2 appears after Str1 in the Searchtring. Else it returns 2.

Create a class **Main** which would get 3 Strings as input and call the static method **stringFinder** present in the UserMainCode.

Input and Output Format:
Input consists of 3 strings.
The first input corresponds to the SearchString.
The second input corresponds to Str1.
The third input corresponds to Str2.
Output consists of a string that is either “yes” or “no”

Sample Input 1:
geniousRajKumarDev
Raj
Dev

Sample Output 1:
yes

Sample Input 2:
geniousRajKumarDev
Dev
Raj

Sample Output 2:
no

```
public class User {
	public static int stringFinder(String str,String s1,String s2)
	{
		int res=0;
		if(str.contains(s1)&&str.contains(s2))
		{
			if(str.indexOf(s1)<str.indexOf(s2))
				res=1;
			else
				res=0;
		}
		return res;
	}
}
```

66) Phone Number Validator

Given a phone number as a string input, write a program to verify whether the phone number is valid using the following business rules:
-It should contain only numbers or dashes (-)
- dashes may appear at any position
-Should have exactly 10 digits

Include a class **UserMainCode** with a static method "**validatePhoneNumber**" that accepts a String input and returns a integer. The method returns 1 if the phone number is valid. Else it returns 2.

Create a class **Main** which would get a String as input and call the static method **validatePhoneNumber** present in the UserMainCode.

Input and Output Format:
Input consists of a string.
Output consists of a string that is either 'Valid' or 'Invalid'

Sample Input 1:
265-265-7777

Sample Output 1:
Valid

Sample Input 2:
265-65-7777

Sample Output 1:
Invalid

```
public class User {
	public static int validatePhoneNumber(String s)
	{
		int res=0;
		if(s.matches("[0-9]{3}(-)[0-9]{3}(-)[0-9]{4}"))
			res=1;
		else
			res=-1;
		return res;
	}
}
```

68) Month : Number of Days

Given two inputs year and month (Month is coded as: Jan=0, Feb=1 ,Mar=2 ...), write a program to find out total number of days in the given month for the given year.

Include a class **UserMainCode** with a static method "**getNumberOfDays**" that accepts 2 integers as arguments and returns an integer. The first argument corresponds to the year and the second argument corresponds to the month code. The method returns an integer corresponding to the number of days in the month.

Create a class **Main** which would get 2 integers as input and call the static method **getNumberOfDays** present in the UserMainCode.

Input and Output Format:
Input consists of 2 integers that correspond to the year and month code.
Output consists of an integer that correspond to the number of days in the month in the given year.

Sample Input:
2000
1

Sample Output:
29

```
public class User {
	public static int getNumberOfDays(int year,int month)
	{
		GregorianCalendar gc=new GregorianCalendar(year,month,1);

	   int days=gc.getActualMaximum(Calendar.DAY_OF_MONTH);
		return days;
	}
}
```

69) Negative String

Given a string input, write a program to replace every appearance of the word "is" by "is not".
If the word "is" is immediately preceeded or followed by a letter no change should be made to the string .

Include a class **UserMainCode** with a static method "**negativeString**" that accepts a String arguement and returns a String.

Create a class **Main** which would get a String as input and call the static method **negativeString** present in the UserMainCode.

Input and Output Format:
Input consists of a String.
Output consists of a String.

Sample Input 1:
This is just a misconception

Sample Output 1:
This is not just a misconception

Sample Input 2:
Today is misty

Sample Output 2:
Today is not misty

```
public class User {
      public static String validateNumber(String s)
      {
     StringTokenizer st=new StringTokenizer(s," ");
     StringBuffer sb=new StringBuffer();
while(st.hasMoreTokens())
{
      String r=st.nextToken();
      if(r.equals("is"))
      {
            sb.append(r.replace("is", "is not"));
      }
      else
            sb.append(r);
      sb.append(" ");
}

// sb.deleteCharAt((sb.length()-1));

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

70) Validate Number

Given a negative number as string input, write a program to validate the number and to print the corresponding positive number.

Include a class **UserMainCode** with a static method "**validateNumber**" that accepts a string argument and returns a string. If the argument string contains a valid negative number, the method returns the corresponding positive number as a string. Else the method returns -1.

Create a class **Main** which would get a String as input and call the static method **validateNumber** present in the UserMainCode.

Input and Output Format:
Input consists of a String.
Output consists of a String.

Sample Input 1:
-94923

Sample Output 1:
94923

Sample Input 2:
-6t

Sample Output 2:
-1

```
public class User {
	public static String validateNumber(String s)
	{
	String res=null;
	int count=0;
	for(int i=1;i<s.length();i++)
	{
	char c=s.charAt(i);
	if(Character.isDigit(c))
	count++;
	}
	if(count==s.length()-1)
	{
		res=String.valueOf(Math.abs(Integer.parseInt(s)));
	}
	else
	    res="-1";
	return res;


	}
}
```