1.Check Sum of Odd Digits

Write a program to read a number , calculate the sum of odd digits (values) present in the given number.

Include a class **UserMainCode** with a static method **checkSum** which accepts a positive integer . The return type should be 1 if the sum is odd . In case the sum is even return -1 as output.

Create a class **Main** which would get the input as a positive integer and call the static method **checkSum** present in the UserMainCode.

Input and Output Format:
Input consists of a positive integer n.
Refer sample output for formatting specifications.

Sample Input 1:

56895

Sample Output 1:

Sum of odd digits is odd.

Sample Input 2:

84228

Sample Output 2:

Sum of odd digits is even.

```
public class UserMainCode {
	public static int SumOfOddsAndEvens(int n){
		int n1,n2=0,n3;
		while(n!=0)
		{
			n1=n%10;
			if((n1%2)!=0)
				n2+=n1;
			n/=10;
		}
		if(n2%2==0)
			n3=-1;
```

```
                else
                        n3=1;


                return n3;
        }
        public static void main(String[] args) {
                int n=84882;
                System.out.println(SumOfOddsAndEvens(n));
        }
}
```

2.Number Validation

Write a program to read a string of 10 digit number , check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

Include a class **UserMainCode** with a static method **validateNumber** which accepts a string as input .
The return type of the output should be 1 if the string meets the above specified format . In case the number does not meet the specified format then return -1 as output.

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

Input and Output Format:

Input consists of a string.

Output is a string specifying the given string is valid or not .

Refer sample output for formatting specifications.

Sample Input 1:

123-456-7895

Sample Output 1:

Valid number format

Sample Input 2:

-123-12344322

Sample Output 2:

Invalid number format

```
import java.util.*;
public class Main {
public static void main(String[] args) {
	Scanner s=new Scanner(System.in);
	String pan=s.next();
	int b=panNumberValidation(pan);
	if(b==1)
		System.out.println("valid Pancard Number");
	else
		System.out.println("not a valid credential");
}
public static int panNumberValidation(String input) {
	int b=0;
	if(input.matches("[0-9]{3}[-]{1}[0-9]{3}[-]{1}[0-9]{4}"))
	{b=1;}
	else
		b=0;
	return b;
}
}
```

3.Sum of Squares of Even Digits

Write a program to read a number , calculate the sum of squares of even digits (values) present in the given number.

Include a class **UserMainCode** with a static method **sumOfSquaresOfEvenDigits** which accepts a positive integer . The return type (integer) should be the sum of squares of the even digits.

Create a class **Main** which would get the input as a positive integer and call the static method sumOfSquaresOfEvenDigits present in the UserMainCode.

Input and Output Format:

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

Sample Input 1:

56895

Sample Output 1:

100

```
public class UserMainCode

{



        public static int display(int number){

            int n1=0,n2=0;

            while(number!=0)

            {

            n1=number%10;

            if((n1%2)==0)
```

```
            n2+=n1*n1;
            number/=10;
            }
            return n2;
        }
    }
```

4.Fetching Middle Characters from String

Write a program to read a string of even length and to fetch two middle most characters from the input string and return it as string output.

Include a class **UserMainCode** with a static method **getMiddleChars** which accepts a string of even length as input . The return type is a string which should be the middle characters of the string.

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

Input and Output Format:

Input consists of a string of even length.

Output is a string .

Refer sample output for formatting specifications.

Sample Input 1:

this

Sample Output 1:

hi

Sample Input 1:

Hell

Sample Output 1:

el

```java
import java.util.Scanner;

public class Middle {

	public static void main(String[] args) {

		Scanner sc=new Scanner(System.in);

		String s=sc.nextLine();
		StringBuffer sb=new StringBuffer();
		if(s.length()%2==0)
		{
		sb.append(s.substring(s.length()/2-1,s.length()/2+1));
		//System.out.println(sb.toString());

		}
		System.out.println(sb.toString());
	}

}
```

5.Check Characters in a String

Write a program to read a string and to test whether first and last character are same. The string is said to be be valid if the 1st and last character are the same. Else the string is said to be invalid.

Include a class **UserMainCode** with a static method **checkCharacters** which accepts a string as input .
The return type of this method is an int. Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

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

Input and Output Format:

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

Sample Input 1:

the picture was great

Sample Output 1:

Valid

Sample Input 1:

this

Sample Output 1:

Invalid

```
import java.util.Scanner;
public class Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int res=UserMainCode.checkCharacter(s);
if(res==1)
{
	System.out.println("Valid");

}
else
	System.out.println("Invalid");
	}


	}




public class UserMainCode {
	public static int checkCharacter(String s)

	{
		int res=-1;
		if(s.charAt(0)==s.charAt(s.length()-1))
		{
		res=1;
```

```
            }
                return res;
        }
}
```

6.Forming New Word from a String

Write a program to read a string and a positive integer n as input and construct a string with first n and last n characters in the given string.

Include a class **UserMainCode** with a static method **formNewWord** which accepts a string and positive integer .

The return type of the output should be a string (value) of first n character and last n character.

Create a class **Main** which would get the input as a string and integer n and call the static method **formNewWord**present in the UserMainCode.

Input and Output Format:

Input consists of a string of even length.

Output is a string .

Note: The given string length must be >=2n.

Refer sample output for formatting specifications.

Sample Input 1:

California
3

Sample Output 1:

Calnia

Sample Input 2:

this
1

Sample Output 2:

ts

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

```
public class Main {


        public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int n=sc.nextInt();
System.out.println(UserMainCode.stringChange(s,n));



}
}

public class UserMainCode {
        public static String stringChange(String s,int n)
        {


                StringBuffer sb=new StringBuffer();
           sb.append(s.substring(0,n));
           sb.append(s.substring(s.length()-n));
           return sb.toString();
        }
}
```

7.Reversing a Number

Write a program to read a positive number as input and to get the reverse of the given number and return it as output.

Include a class **UserMainCode** with a static method **reverseNumber** which accepts a positive integer .

The return type is an integer value which is the reverse of the given number.

Create a **Main** class which gets the input as a integer and call the static method **reverseNumber** present in the**UserMainCode**

Input and Output Format:

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

Sample Input 1:

543

Sample Output 1:

345

Sample Input 1:

1111

Sample Output 1:

1111

```
import java.util.Scanner;
public class Main
{

public static void main(String[] args)
{

Scanner sc=new Scanner(System.in);
int a=sc.nextInt();
System.out.println(UserMainCode.reverse(a));

}
}
```

```
public class UserMainCode
```

```
{
        public static int reverse(int a)
        {

        String s=String.valueOf(a);
        StringBuffer sb=new StringBuffer(s);
    sb.reverse();   //reverse return type is void
    int res=Integer.parseInt(sb.toString());
    return res;
        }
}
```

8.Array List Sorting and Merging

Write a code to read two int array lists of size 5 each as input and to merge the two arrayLists, sort the merged arraylist in ascending order and fetch the elements at 2nd, 6th and 8th index into a new arrayList and return the final ArrayList.

Include a class **UserMainCode** with a static method **sortMergedArrayList** which accepts 2 ArrayLists.

The return type is an ArrayList with elements from 2,6 and 8th index position .Array index starts from position 0.

Create a **Main** class which gets two array list of size 5 as input and call the static method **sortMergedArrayList**present in the **UserMainCode.**

Input and Output Format:

Input consists of two array lists of size 5.

Output is an array list .

Note - The first element is at index 0.

Refer sample output for formatting specifications.

Sample Input 1:

3
1
17
11
19
5
2
7
6
20

Sample Output 1:

3

11

19

Sample Input 2:

1

2

3

4

5

6

7

8

9

10

Sample Output 2:

3

7

9

9.Validating Date Format

Obtain a date string in the format dd/mm/yyyy. Write code to validate the given date against the given format.

Include a class **UserMainCode** with a static method **validateDate** which accepts a string .

The return type of the validateDate method is 1 if the given date format matches the specified format , If the validation fails return the output as -1.

Create a **Main** class which gets date string as an input and call the static method **validateDate** present in the**UserMainCode.**

Input and Output Format:

Input is a string .

Refer sample output for formatting specifications

Sample Input 1:

12/06/1987
Sample Output 1:
Valid date format

Sample Input 2:

03/1/1987
Sample Output 2:
Invalid date format

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Qus8Main {

      public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
```

```
sdf.setLenient(false);
int res=0;
if(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))
{

	try {
		Date d=sdf.parse(s);
		res=1;
	} catch (ParseException e) {
		res=-1;
	}

System.out.println(res);

	}
}
}
```

10.Validate Time

Obtain a time string as input in the following format 'hh:mm am' or 'hh:mm pm'. Write code to validate it using the following rules:

- It should be a valid time in 12 hrs format
- It should have case insensitive AM or PM

Include a class **UserMainCode** with a static method **validateTime** which accepts a string.

If the given time is as per the given rules then return 1 else return -1.If the value returned is 1 then print as valid time else print as Invalid time.

Create a **Main** class which gets time(string value) as an input and call the static method **validateTime** present in the**UserMainCode.**

Input and Output Format:

Input is a string .

Output is a string .

Sample Input 1:

09:59 pm
Sample Output 1:
Valid time

Sample Input 2:

10:70 AM
Sample Output 2:
Invalid time

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Qus8Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
SimpleDateFormat sdf=new SimpleDateFormat("hh:mm a");
sdf.setLenient(false);
int res=0;

	try {
		Date d=sdf.parse(s);
		res=1;
	} catch (ParseException e) {
		res=-1;
	}

System.out.println(res);


}
}
```

11.String Encryption

Given an input as string and write code to encrypt the given string using following rules and return the encrypted string:

1. Replace the characters at odd positions by next character in alphabet.
2. Leave the characters at even positions unchanged.

Note:
- If an odd position charater is 'z' replace it by 'a'.
- Assume the first character in the string is at position 1.

Include a class **UserMainCode** with a static method **encrypt** which accepts a string.

The return type of the output is the encrypted string.

Create a **Main** class which gets string as an input and call the static method **encrypt** present in the **UserMainCode.**

Input and Output Format:

Input is a string .

Output is a string.

Sample Input 1:

curiosity
Sample Output 1:
dusipsjtz

Sample Input 2:

zzzz
Sample Output 2:
azaz

```
public class Main {

public static void main(String[] args) {

String s1="zzzz";

System.out.println(stringFormatting(s1));

}

public static String stringFormatting(String s1) {

StringBuffer sb=new StringBuffer();

for(int i=0;i<s1.length();i++){

char c=s1.charAt(i);

if(i%2==0){
```

```
if(c==122)

c=(char) (c-25);

else{

c=(char) (c+1);}

sb.append(c);}

else

sb.append(c);}

return sb.toString();

}

}
```

12.Password Validation

Given a method with a password in string format as input. Write code to validate the password using following rules:

- Must contain at least one digit
- Must contain at least one of the following special characters @, #, $
Length should be between 6 to 20 characters.

Include a class **UserMainCode** with a static method **validatePassword** which accepts a password string as input.

If the password is as per the given rules return 1 else return -1.If the return value is 1 then print valid password else print as invalid password.

Create a **Main** class which gets string as an input and call the static method **validatePassword** present in the**UserMainCode.**

Input and Output Format:

Input is a string .

Output is a string .

Sample Input 1:

%Dhoom%
Sample Output 1:
Invalid password

Sample Input 2:

#@6Don
Sample Output 2:
Valid password

```
public class UserMainCode {

        public static int display(String password){

          if(password.matches(".*[0-9]{1,}.*") && password.matches(".*[@#$]{1,}.*")
&& password.length()>=6 && password.length()<=20)

                {

                        return 1;

                }

                else

                {

                        return -1;

                }

        }
```

}

13.Removing vowels from String

Given a method with string input. Write code to remove vowels from even position in the string.

Include a class **UserMainCode** with a static method **removeEvenVowels** which accepts a string as input.

The return type of the output is string after removing all the vowels.

Create a **Main** class which gets string as an input and call the static method **removeEvenVowels** present in the**UserMainCode.**

Input and Output Format:

Input is a string .

Output is a string .

Assume the first character is at position 1 in the given string.

Sample Input 1:

commitment
Sample Output 1:
cmmitmnt

Sample Input 2:

capacity
Sample Output 2:
Cpcty

```
public class Main {

	public static void main(String[] args) {

		String s1="capacity";
```

```
		System.out.println(removeEvenElements(s1));
	}
	public static String removeEvenElements(String s1) {
		StringBuffer sb1=new StringBuffer();
		for(int i=0;i<s1.length();i++)
			if((i%2)==0)
				sb1.append(s1.charAt(i));
			else if((i%2)!=0)
				if(s1.charAt(i)!='a' && s1.charAt(i)!='e' && s1.charAt(i)!='i' &&
s1.charAt(i)!='o' && s1.charAt(i)!='u')
					if(s1.charAt(i)!='A' && s1.charAt(i)!='E' &&
s1.charAt(i)!='I' && s1.charAt(i)!='O' && s1.charAt(i)!='U')
					 sb1.append(s1.charAt(i));
		return sb1.toString();
	}
}
```

14.Sum of Powers of elements in an array

Given a method with an int array. Write code to find the power of each individual element accoding to its position index, add them up and return as output.

Include a class **UserMainCode** with a static method **getSumOfPower** which accepts an integer array as input.

The return type of the output is an integer which is the sum powers of each element in the array.

Create a **Main** class which gets integer array as an input and call the static method **getSumOfPower** present in the**UserMainCode.**

Input and Output Format:

Input is an integer array.First element corresponds to the number(n) of elements in an array.The next inputs corresponds to each element in an array.

Output is an integer .

Sample Input 1:

4
3
6
2
1

Sample Output 1:

12

Sample Input 2:

4
5
3
7
2

Sample Output 2:

61

```
public class useerm{

public static int display(int n,int[]a)

{

                    {

                            int sum=0;

                            for(int i=0;i<n;i++)
```

```
                    sum=(int)(sum+Math.pow(a[i], i));

                return sum;

    }}}
```

15.Difference between largest and smallest elements in an array

Given a method taking an int array having size more than or equal to 1 as input. Write code to return the difference between the largest and smallest elements in the array. If there is only one element in the array return the same element as output.

Include a class **UserMainCode** with a static method **getBigDiff** which accepts a integer array as input.

The return type of the output is an integer which is the difference between the largest and smallest elements in the array.

Create a **Main** class which gets integer array as an input and call the static method **getBigDiff** present in the**UserMainCode.**

Input and Output Format:

Input is an integer array.First element in the input represents the number of elements in an array.

Size of the array must be >=1

Output is an integer which is the difference between the largest and smallest element in an array.

Sample Input 1:

4
3
6
2
1

Sample Output 1:

5

Sample Input 2:

4
5
3

7
2

Sample Output 2:
5

```
import java.util.Arrays;

public class kape1 {

public static int display(int []array)



{

	Arrays.sort(array);

	int n=array[array.length-1]-array[0];

	int b=array.length;

	if(b==1)

	{

		n=array[0];

	}

	return n;

	}

}
```

16.Find the element position in a reversed string array

Given a method with an array of strings and one string variable as input. Write code to sort the given array in reverse alphabetical order and return the postion of the given string in the array.

Include a class **UserMainCode** with a static method **getElementPosition** which accepts an array of strings and a string variable as input.

The return type of the output is an integer which is the position of given string value from the array.

Create a **Main** class which gets string array and a string variable as an input and call the static method**getElementPosition** present in the **UserMainCode.**

Input and Output Format:

Input is an string array. First element in the input represents the size the array

Assume the position of first element is 1.

Output is an integer which is the position of the string variable

Sample Input 1:
4
red
green
blue
ivory
ivory

Sample Output 1:
2

Sample Input 2:

3
grape
mango
apple
apple

Sample Output 2:
3

17.Generate the series

Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:
1+3-5+7-9…+/-n.

Include a class **UserMainCode** with a static method **addSeries** which accepts a positive integer .

The return type of the output should be an integer .

Create a class **Main** which would get the input as a positive integer and call the static method **addSeries** present in the UserMainCode.

Input and Output Format:

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

Sample Input 1:

9

Sample Output 1:

-3

Sample Input 2:

11

Sample Output 2:

8

```
public class UserMainCode
{
	public static int generateSeries(int n)
	{
		int i=0,sumo=0,sume=0,sum=1;
		if(n==1)
		{
			sum=n;
			break;
		}


for(i=3;i<=n;i=i+4)
{


		sumo=sumo+i;
}
for(i=5;i<=n;i=i+4)
{
	sume=sume+i;
}
```

```
sum+=sumo-sume;
return sum;
	}
	}
```

18.Calculate Electricity Bill

Given a method calculateElectricityBill() with three inputs. Write code to calculate the current bill.

Include a class **UserMainCode** with a static method **calculateElectricityBill** which accepts 3 inputs .The return type of the output should be an integer .

Create a class **Main** which would get the inputs and call the static method **calculateElectricityBill** present in the UserMainCode.

Input and Output Format:

Input consist of 3 integers.
First input is previous reading, second input is current reading and last input is per unit charge.
Reading Format - XXXXXAAAAA where XXXXX is consumer number and AAAAA is meter reading.
Output is a single integer corresponding to the current bill.

Refer sample output for formatting specifications.

Sample Input 1:

ABC2012345
ABC2012660
4

Sample Output 1:

1260

Sample Input 2:

ABCDE11111

ABCDE11222

3

Sample Output 2:

333

```
import java.util.Scanner;
public class Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.nextLine();
int c=sc.nextInt();
System.out.println(UserMainCode.calculateElectricityBill(s1,s2, c));



}
}
```

```
public class UserMainCode {
	public static int calculateElectricityBill(String s1,String s2,int c)

	{

		int a=Integer.parseInt(s1.substring(5));
		int b=Integer.parseInt(s2.substring(5));
		int res=Math.abs((b-a)*c);
		return res;
	}
}
```

19.Sum of Digits in a String

Write code to get the sum of all the digits present in the given string.

Include a class **UserMainCode** with a static method **sumOfDigits** which accepts string input.

Return the sum as output. If there is no digit in the given string return -1 as output.

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

Input and Output Format:

Input consists of a string.

Output is a single integer which is the sum of digits in a given string.

Refer sample output for formatting specifications.

Sample Input 1:

good23bad4

Sample Output 1:

9

Sample Input 2:

good

Sample Output 2:

-1

```
public class Main {

public static void main(String[] args) {

String s1="goodbad";

getvalues(s1);

}

public static void getvalues(String s1) {

int sum=0;

for(int i=0;i<s1.length();i++)

{
```

```
char a=s1.charAt(i);

if(Character.isDigit(a))

{

int b=Integer.parseInt(String.valueOf(a));

sum=sum+b;

}

}

if(sum==0)

{

System.out.println(-1);

}

else

System.out.println(sum);

}

}
```

20.String Concatenation

Write code to get two strings as input and If strings are of same length simply append them together and return the final string. If given strings are of different length, remove starting characters from the longer string so that both strings are of same length then append them together and return the final string.

Include a class **UserMainCode** with a static method **concatstring** which accepts two string input.

The return type of the output is a string which is the concatenated string.

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

Input and Output Format:

Input consists of two strings.

Output is a string.

Refer sample output for formatting specifications.

Sample Input 1:

Hello
hi

Sample Output 1:

lohi

Sample Input 2:

Hello

Delhi

Sample Output 2:

HelloDelhi

```
import java.util.Scanner;
public class Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.nextLine();

System.out.println(UserMainCode.concatString(s1,s2));



}
}



public class UserMainCode
{
	public static String concatString(String s1,String s2)
	{
```

```
        StringBuffer sb=new StringBuffer();
        int n=s2.length();
        sb.append(s1.substring(s1.length()-n));
        sb.append(s2.substring(0));
        return sb.toString();
        }
}
```

21.Color Code

Write a program to read a string and validate whether the given string is a valid color code based on the following rules:

- Must start with "#" symbol
- Must contain six characters after #
- It may contain alphabets from A-F or digits from 0-9

Include a class **UserMainCode** with a static method **validateColorCode** which accepts a string. The return type (integer) should return 1 if the color is as per the rules else return -1.

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

Input and Output Format:

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

Sample Input 1:

#FF9922
Sample Output 1:
Valid

Sample Input 2:

#FF9(22
Sample Output 2:

Invalid

```java
import java.util.Scanner;


public class Main {


	public static void main(String[] args)
	{
	Scanner sc=new Scanner(System.in);
	String s=sc.nextLine();
	//UserMainCode u=new UserMainCode();
	int b=UserMainCode.validateColorCode(s);
	if(b==1)
	{
		System.out.println("Valid color code");

	}
	else
	{
		System.out.println("Invalid color code ");
	}

	}

}
public class UserMainCode
{
public static int validateColorCode(String a)
{
	int r=-1;
	if(a.matches("(#)[A-F0-9]{6}"))

	{
		r=1;
	}
	return r;
}
}
```

22.Three Digits

Write a program to read a string and check if the given string is in the format "CTS-XXX" where XXX is a three digit number.
Include a class **UserMainCode** with a static method **validatestrings** which accepts a string. The return type (integer) should return 1 if the string format is correct else return -1.

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

Input and Output Format:

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

Sample Input 1:

CTS-215
Sample Output 1:

Valid

Sample Input 2:

CTS-2L5
Sample Output 2:

Invalid

****
CTS-215

```
public class Main {

public static void main(String[] args) {

String s1="CTS-2j4";

getvalues(s1);

}
```

```
public static void getvalues(String s1) {

if(s1.matches("(CTS)[-]{1}[0-9]{3}"))

{

System.out.println(1);

}

else

System.out.println(-1);

}

}
```

23.Removing Keys from HashMap

Given a method with a HashMap<Integer,string> as input. Write code to remove all the entries having keys multiple of 4 and return the size of the final hashmap.

Include a class **UserMainCode** with a static method **sizeOfResultandHashMap** which accepts hashmap as input.

The return type of the output is an integer which is the size of the resultant hashmap.

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

Input and Output Format:

First input corresponds to the size of the hashmap.

Input consists of a hashmap<integer,string>.

Output is an integer which is the size of the hashmap.

Refer sample output for formatting specifications.

Sample Input 1:

3
2
hi

4
hello
12
hello world

Sample Output 1:

1

Sample Input 2:

3

2

hi

4

sdfsdf

3

asdf

Sample Output 2:

2

24.Largest Element

Write a program to read an int array of odd length, compare the first, middle and the last elements in the array and return the largest. If there is only one element in the array return the same element.

Include a class **UserMainCode** with a static method **checkLargestAmongCorner** which accepts an int arrayThe return type (integer) should return the largest element among the first, middle and the last elements.

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

Assume maximum length of array is 20.

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

5

2

3

8

4

5

Sample Output 1:

8

```
import java.util.Scanner;
public class Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
int n=sc.nextInt();

int a[]=new int[n];
for(int i=0;i<n;i++)
{
	a[i]=sc.nextInt();
}
System.out.println(UserMainCode.checkLargestAmongCorner(a));



}
}
```

```
public class UserMainCode
{
	public static int checkLargestAmongCorner(int a[])
	{
		int max=0;
	int m=a[a.length/2];
	int f=a[0];
	int l=a[a.length-1];
	if(m>f && m>l)
	{
	  max=m;
	}
	else if(f>m && f>l)
	{
	  max=f;
	}
	else
```

```
  max=l;

return max;
}
}
```

25.nCr

Write a program to calculate the ways in which r elements can be selected from n population, using nCr formula nCr=n!/r!(n-r)! where first input being n and second input being r.

Note1 : n! factorial can be achieved using given formula n!=nx(n-1)x(n-2)x..3x2x1.
Note2 : 0! = 1.
Example 5!=5x4x3x2x1=120

Include a class **UserMainCode** with a static method **calculateNcr** which accepts two integers. The return type (integer) should return the value of nCr.

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

Input and Output Format:

Input consists of 2 integers. The first integer corresponds to n, the second integer corresponds to r.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

4
3

Sample Output 1:

4

```
import java.util.Scanner;
public class Main {


      public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int r=sc.nextInt();

System.out.println(UserMainCode.calculateNcr(n,r) );
```

```
}
}
```

```
public class UserMainCode
{
	public static int calculateNcr(int n,int r)
	{
		int i,prod=1,prod1=1,prod2=1;
	for(i=1;i<=n;i++)
	{
		 prod=prod*i;

	}
		for(i=1;i<=r;i++)
		{
			prod1=prod1*i;
		}
		int diff=n-r;
		for(i=1;i<=diff;i++)
		{
			prod2=prod2*i;
		}
		int dem=prod1*prod2;
		int res=prod/dem;
		return res;
	}
	}
```

26.Sum of Common Elements

Write a program to find out sum of common elements in given two arrays. If no common elements are found print - “No common elements”.
Include a class **UserMainCode** with a static method **getSumOfIntersection** which accepts two integer arrays and their sizes. The return type (integer) should return the sum of common elements.

Create a Class Main which would be used to accept 2 Input arrays and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 2+m+n integers. The first integer corresponds to m (Size of the 1st array), the second integer corresponds to n (Size of the 2nd array), followed by m+n integers corresponding to the array elements.

Output consists of a single Integer corresponds to the sum of common elements or a string “No common elements”.

Refer sample output for formatting specifications.

Assume the common element appears only once in each array.

Sample Input 1:

4
3
2
3
5
1
1
3
9

Sample Output 1:

4

Sample Input 2:

4
3
2
3
5
1
12
31
9

Sample Output 2:

No common elements

public class Main {

public static void main(String[] args)

```
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int m=sc.nextInt();
int[] a=new int[n];
int[] b=new int[m];
for(int i=0;i<n;i++)
a[i]=sc.nextInt();
for(int i=0;i<m;i++)
b[i]=sc.nextInt();
int u=UserMainCode.display(a,b);
if(u==-1)
System.out.println("No common elements");
else
System.out.println(u);}}


public class UserMainCode {
public static int display(int a[],int b[])
{


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

```
{

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

{if(a[i]==b[j])

sum=sum+a[i];

}}

if(sum==0)

return -1;

else

return sum;

}}
```

27.Validating Input Password

102.Write a code get a password as string input and validate using the rules specified below. Apply following validations:

1. Minimum length should be 8 characters
2. Must contain any one of these three special characters @ or _ or #
3. May contain numbers or alphabets.
4. Should not start with special character or number
5. Should not end with special character

Include a class **UserMainCode** with a static method **validatePassword** which accepts password string as input and returns an integer. The method returns 1 if the password is valid. Else it returns -1.

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

Input and Output Format:

Input consists of a string.

Output is a string Valid or Invalid.

Refer sample output for formatting specifications.

Sample Input 1:

ashok_23

Sample Output 1:

Valid

Sample Input 2:

1980_200

Sample Output 2:

Invalid

```
public class UserMainCode {
    public static int validatePassword(String password){
        String regEx = "[^0-9|@|_|#](.)*[@|_|#](.)*[^@|_|#]";
        if(password.length()>=8 && password.matches(regEx)){
            return 1;
        }
        return -1;
    }
}
```

```
if(s.length()>=8&&s.matches("[^0-9|#|_|@](.*)[#|_|@](.*)[^@|_|#]"))
{
      System.out.println("valid");
}
```

28.ID Validation

Write a program to get two string inputs and validate the ID as per the specified format.

Include a class **UserMainCode** with a static method **validateIDLocations** which accepts two strings as input.

The return type of the output is a string Valid Id or Invalid Id.

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

Input and Output Format:

Input consists of two strings.

First string is ID and second string is location. ID is in the format CTS-LLL-XXXX where LLL is the first three letters of given location and XXXX is a four digit number.

Output is a string Valid id or Invalid id.

Refer sample output for formatting specifications.

Sample Input 1:

CTS-hyd-1234
hyderabad
Sample Output 1:

Valid id

Sample Input 2:

CTS-hyd-123
hyderabad
Sample Output 2:

Invalid id

```
import java.util.*;

public class Main {

public static void main(String[] args) {

String s1="CTS-hyd-1234";

String s2="hyderabad";

boolean b=formattingString(s1,s2);

if(b==true)
```

```
System.out.println("String format:CTS-LLL-XXXX// valid id");

else

System.out.println("not in required format");

}

public static boolean formattingString(String s1, String s2) {

String s3=s2.substring(0, 3);

boolean b=false;

StringTokenizer t=new StringTokenizer(s1,"-");

String s4=t.nextToken();

String s5=t.nextToken();

String s6=t.nextToken();

if(s4.equals("CTS") && s5.equals(s3) && s6.matches("[0-9]{4}"))

b=true;

else{

b=false;}

return b;

}

}
```

29.Remove Elements

Write a program to remove all the elements of the given length and return the size of the final array as output. If there is no element of the given length, return the size of the same array as output.

Include a class **UserMainCode** with a static method **removeElements** which accepts a string array, the number of elements in the array and an integer. The return type (integer) should return the size of the final array as output.

Create a Class Main which would be used to accept Input String array and a number and call the static method present in UserMainCode.

Assume maximum length of array is 20.

Input and Output Format:

Input consists of a integers that corresponds to n, followed by n strings and finally m which corresponds to the length value.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:
5
a
bb
b
ccc
ddd
2
Sample Output 1:
4

import java.util.*;

public class Main

{

public static void main(String[] args)

{

```
Scanner sc=new Scanner(System.in);

int n=sc.nextInt();
sc.nextLine();
```

```
String[] a=new String[n];
for(int i=0;i<n;i++)
 a[i]=sc.nextLine();

int m=sc.nextInt();
```

System.out.println(UserMainCode.display(a,m));

}}

```
import java.util.*;


public class UserMainCode
{


    public static int display(String[] a,int m){


            int u=a.length;

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

            {

            if(a[i].length()==m)

            u--;

            }

            return u;

            }}
```

30.Find the difference between Dates in months

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

Include a class **UserMainCode** with a static method **getMonthDifference** which accepts two date strings as input.

The return type of the output is an integer which returns the diffenece between two dates in months.

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

Input and Output Format:

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

2012-03-01

2012-04-16

Sample Output 1:

1

Sample Input 2:

2011-03-01
2012-04-16

Sample Output 2:

13

```
import java.text.*;

import java.util.*;
```

```
public class Main {
public static void main(String[] args) throws ParseException {
String s1="2012-03-01";
String s2="2012-03-16";
System.out.println(monthsBetweenDates(s1,s2));
}
public static int monthsBetweenDates(String s1, String s2) throws ParseException {
SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
Date d1=sdf.parse(s1);
Date d2=sdf.parse(s2);
Calendar cal=Calendar.getInstance();

cal.setTime(d1);
int months1=cal.get(Calendar.MONTH);
int year1=cal.get(Calendar.YEAR);
cal.setTime(d2);
int months2=cal.get(Calendar.MONTH);
int year2=cal.get(Calendar.YEAR);
int n=((year2-year1)*12)+(months2-months1);
return n;
}
}
```

31.Sum of cubes and squares of elements in an array

Write a program to get an int array as input and identify even and odd numbers. If number is odd get cube of it, if number is even get square of it. Finally add all cubes and squares together and return it as output.

Include a class **UserMainCode** with a static method **addEvenOdd** which accepts integer array as input.

The return type of the output is an integer which is the sum of cubes and squares of elements in the array.

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

Input and Output Format:

Input consists of integer array.

Output is an integer sum.

Refer sample output for formatting specifications.

Sample Input 1:

5

2

6

3

4

5

Sample Output 1:

208

```
public class Main {

public static void main(String[] args) {
```

```
int a[]={2,4,3,5,6};

System.out.println(summationPattern(a));

}

public static int summationPattern(int[] a) {

int n1=0,n2=0;

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

if(a[i]%2==0)

n1+=(a[i]*a[i]);

else

n2+=(a[i]*a[i]*a[i]);

return n1+n2;

}

}
```

32.IP Validator

Write a program to read a string and validate the IP address. Print “Valid” if the IP address is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **ipValidator** which accepts a string. The return type (integer) should return 1 if it is a valid IP address else return 2.

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

Input and Output Format:

Input consists of a string that corresponds to an IP.

Output consists of a string(“Valid” or “Invalid”).

Refer sample output for formatting specifications.

Note: An IP address has the format a.b.c.d where a,b,c,d are numbers between 0-255.

Sample Input 1:
132.145.184.210
Sample Output 1:
Valid

Sample Input 2:
132.145.184.290
Sample Output 2:
Invalid

```
import java.util.*;

public class Main {
	public static void main(String[] args) {
		String ipAddress="10.230.110.160";
	   boolean b=validateIpAddress(ipAddress);
	   if(b==true)
		System.out.println("valid ipAddress");
	   else
		System.out.println("not a valid ipAddress");
	}
	public static boolean validateIpAddress(String ipAddress) {
		boolean b1=false;
		StringTokenizer t=new StringTokenizer(ipAddress,".");
                String s=t.nextToken();
               int a=Integer.parseInt(s);
```

```
                                                    int
b=Integer.parseInt(t.nextToken());

            int c=Integer.parseInt(t.nextToken());

            int d=Integer.parseInt(t.nextToken());

            if((a>=0 && a<=255)&&(b>=0 && b<=255)&&(c>=0 && c<=255)&&(d>=0
&& d<=255))

                  b1=true;

            return b1;

      }

}
```

33.Difference between two dates in days

Get two date strings as input and write code to find difference between two dates in days.

Include a class **UserMainCode** with a static method **getDateDifference** which accepts two date strings as input.

The return type of the output is an integer which returns the diffenece between two dates in days.

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

Input and Output Format:

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

2012-03-12

2012-03-14

Sample Output 1:

2

Sample Input 2:

2012-04-25
2012-04-28

Sample Output 2:

3

```
import java.text.*;

import java.util.*;

public class Main {

public static int dateDifference(String s1,String s2) throws ParseException{

        SimpleDateFormat sd=new SimpleDateFormat("yyyy-MM-dd");

        Date d=sd.parse(s1);

        Calendar c=Calendar.getInstance();

        c.setTime(d);

        long d1=c.getTimeInMillis();

        d=sd.parse(s2);

        c.setTime(d);

        long d2=c.getTimeInMillis();

        int n=Math.abs((int) ((d1-d2)/(1000*3600*24)));

        return n;

        }

public static void main(String[] args) throws ParseException {
```

```
        String s1="2012-03-12";

        String s2="2012-03-14";

        System.out.println(dateDifference(s1,s2));

        }

}
```

34.File Extension

Write a program to read a file name as a string and find out the file extension and return it as output. For example, the file sun.gif has the extension gif.

Include a class **UserMainCode** with a static method **fileIdentifier** which accepts a string. The return type (string) should return the extension of the input string (filename).

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

Input and Output Format:

Input consists of a string that corresponds to a file name.

Output consists of a string(extension of the input string (filename)).

Refer sample output for formatting specifications.

Sample Input 1:

sun.gif

Sample Output 1:

Gif

```
import java.util.*;

public class Main {

        public static String extensionString(String s1){

                StringTokenizer t=new StringTokenizer(s1,".");

                String ss=t.nextToken();

                String s2=t.nextToken();
```

```
        return s2;

    }

    public static void main(String[] args) {

        String s1="sun.gif";

        System.out.println(extensionString(s1));

    }

}
```

35.Find common characters and unique characters in string

Given a method with two strings as input. Write code to count the common and unique letters in the two strings.

Note:
- Space should not be counted as a letter.
- Consider letters to be case sensitive. ie, "a" is not equal to "A".

Include a class **UserMainCode** with a static method **commonChars** which accepts two strings as input.

The return type of the output is the count of all common and unique characters in the two strings.

Create a class **Main** which would get the inputs and call the static method **commonChars** present in the UserMainCode.

Input and Output Format:

Input consists of two strings.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

a black cow

battle ship

Sample Output 1:

2

[**Explanation** : b, l and a are the common letters between the 2 input strings. But 'a' appears more than once in the 1st string. So 'a' should not be considered while computing the count value.]

Sample Input 2:

australia
sri lanka

Sample Output 2:

4

```
import java.util.Arrays;

import java.util.StringTokenizer;

public class PO

{

public static int display(String s,String s1)

{

int c=0,m=0;String t=null;

char a[]=s.toCharArray();

char b[]=s1.toCharArray();

Arrays.sort(a);

Arrays.sort(b);

s=new String(a);

s1=new String(b);

StringTokenizer st=new StringTokenizer(s);

StringTokenizer st1=new StringTokenizer(s1);

s=st.nextToken();
```

```
s1=st1.nextToken();

if(s.length()>s1.length())

{t=s1;

s1=s;

s=t;

}

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

{

for(int j=0;j<s1.length();j++)

{

if(s.charAt(i)==s1.charAt(j))

{

if((s.indexOf(s.charAt(i))==s.lastIndexOf(s.charAt(i)))&&(s1.indexOf(s1.charAt(j))==s1.lastInd
exOf(s1.charAt(j))))

{

c++;

}

}}}

return c;



}

}
```

36.)Initial Format

Write a program to input a person's name in the format "FirstName LastName" and return the person name in the following format - "LastName, InitialOfFirstName".

Include a class **UserMainCode** with a static method **nameFormatter** which accepts a string. The return type (string) should return the expected format.

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

Input and Output Format:

Input consists of a string that corresponds to a Person's name.

Output consists of a string(person's name in expected format).

Refer sample output for formatting specifications.

Sample Input :
Jessica Miller
Sample Output:
Miller, J

```
import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

String s1="vishal jadiya";

getvalues(s1);

}

public static void getvalues(String s1) {

StringBuffer sb=new StringBuffer();

StringTokenizer st=new StringTokenizer(s1," ");

String s2=st.nextToken();
```

String s3=st.nextToken();

```
sb.append(s3);

 sb.append(",");
```

sb.append(s2.substring(0,1).toUpperCase());

System.out.println(sb);

}

}

37) Character cleaning

Write a program to input a String and a character, and remove that character from the given String. Print the final string.
Include a class **UserMainCode** with a static method **removeCharacter** which accepts a string and a character. The return type (string) should return the character cleaned string.

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

Input and Output Format:

Input consists of a string and a character.

Output consists of a string(the character cleaned string).

Refer sample output for formatting specifications.

Sample Input :
elephant
e
Sample Output:
lphant

```
import java.util.Scanner;
public class Qus8Main {


        public static void main(String[]args){
```

```
Scanner sc=new Scanner(System.in);
String name=sc.nextLine();
     char ch=sc.nextLine().charAt(0);
StringBuffer sb=new StringBuffer(name);
     for(int i=0;i<sb.length();i++)
     {if(ch==sb.charAt(i))
{
sb.deleteCharAt(i);
i--;
}
}
System.out.print(sb.toString());
}}
```

38) Vowel Check

Write a program to read a String and check if that String contains all the vowels. Print “yes” if the string contains all vowels else print “no”.
Include a class **UserMainCode** with a static method **getVowels** which accepts a string. The return type (integer) should return 1 if the String contains all vowels else return -1.

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

Input and Output Format:

Input consists of a string.

Output consists of a string(“yes” or “no”).

Refer sample output for formatting specifications.

Sample Input 1:
abceiduosp
Sample Output 1:
yes

Sample Input 2:
bceiduosp
Sample Output 2:
No

```
import java.util.Scanner;

public class Qus8Main {


            public static void main(String[] name)
            {
                  Scanner sc=new Scanner(System.in);
                  String s=sc.nextLine();
                  System.out.println(Qus8.display(s));
            }
}


public class Qus8 {
      public static int display(String name){

            String s1=name;
      int n1=0,n2=0,n3=0,n4=0,n5=0;
      for(int i=0;i<s1.length();i++){
      char c=s1.charAt(i);
      if(c=='a' || c=='A')
      n1++;
      if(c=='e' || c=='E')
      n2++;
      if(c=='i' || c=='I')
      n3++;
      if(c=='o' || c=='O')
      n4++;
      if(c=='u' || c=='U')
      n5++;}
      if(n1==1 && n2==1 && n3==1 && n4==1 && n5==1)
      return 1;
      else
      return 0 ;
      }

      }
```

39) Swap Characters

Write a program to input a String and swap the every 2 characters in the string. If size is an odd number then keep the last letter as it is. Print the final swapped string.
Include a class **UserMainCode** with a static method **swapCharacter** which accepts a string. The return type (String) should return the character swapped string.

Create a Class Main which would be used to accept Input 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:
TRAINER
Sample Output 1:
RTIAENR

Sample Input 2:
TOM ANDJERRY
Sample output 2:
OT MNAJDREYR

```
import java.util.Scanner;

public class Qus8Main {


	public static void main(String[] args) {
	String s1="TRAINER";
	getvalues(s1);
	}
	public static void getvalues(String s1)
	{
	StringBuffer sb=new StringBuffer();
	int l=s1.length();
	if(l%2==0)
	{
	for(int i=0;i<s1.length()-1;i=i+2)
	{
	char a=s1.charAt(i);
	char b=s1.charAt(i+1);
	sb.append(b)
	   sb.append(a);
	}
	System.out.println(sb);
	}
	else
	{
	for(int i = 0;i<s1.length()-1;i=i+2)
	{
	char a=s1.charAt(i);
	char b=s1.charAt(i+1);
	sb.append(b).append(a);
	}
	sb.append(s1.charAt(l-1));
	System.out.println(sb);
	}
	}

	}
```

40) Average of Elements in Hashmap

Given a method with a HashMap<int, float> as input. Write code to find out avg of all values whose keys are even numbers. Round the average to two decimal places and return as output.

[Hint : If the average is 5.901, the rounded average value is 5.9 . It the average is 6.333, the rounded average value is 6.33 .]

Include a class **UserMainCode** with a static method **avgOfEven** which accepts a HashMap<int, float> as input.

The return type of the output is a floating point value which is the average of all values whose key elements are even numbers.

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

Input and Output Format:

Input consists of the number of elements in the HashMap and the HashMap<int, float>.

Output is a floating point value that corresponds to the average.

Refer sample output for formatting specifications.

Sample Input 1:

3

1

2.3

2

4.1

6

6.2

Sample Output 1:

5.15

Sample Input 2:

3

9
3.1
4
6.3
1
2.6

Sample Output 2:

6.3

41)Calculate Average – Hash Map

Write amethod that accepts the input data as a hash map and finds out the avg of all values whose keys are odd numbers.

Include a class **UserMainCode** with a static method **calculateAverage** which accepts a HashMap<Integer,Double> and the size of the HashMap. The return type (Double) should return the calculated average. Round the average to two decimal places and return it.

Create a Class Main which would be used to accept Input values and store it as a hash map, and call the static method present in UserMainCode.

Input and Output Format:

Input consists of an integer n corresponds to number of hash map values, followed by 2n values. (index followed by value).

Output consists of a Double.

Refer sample input and output for formatting specifications.

Sample Input :

4
1
3.41
2
4.1
3
1.61
4
2.5

Sample Output :

2.51

42) Count Sequential Characters

109.Get a string as input and write code to count the number of characters which gets repeated 3 times consecutively and return that count (ignore case). If no character gets repeated 3 times consecutively return -1.

Include a class **UserMainCode** with a static method **countSequentialChars** which accepts a string as input.

The return type of the output is the repeat count.

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

Input and Output Format:

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

abcXXXabc

Sample Output 1:

1

Sample Input 2:

aaaxxyzAAAx

Sample Output 2:

2

```
public class Main {

public static void main(String[] args) {

String input1="aaxxyzAAx";

System.out.println(consecutiveRepeatitionOfChar(input1));

}
```

```
public static int consecutiveRepeatitionOfChar(String input1) {

int c=0;

int n=0;

for(int i=0;i<input1.length()-1;i++){

if(input1.charAt(i)==input1.charAt(i+1))

n++;

else

n=0;

if(n==2)

c++; }

return c;

}

}
```

43) Length of the Largest Chunk

Write a program to read a string and find the length of the largest chunk in the string. If there are no chunk print “No chunks” else print the length.
NOTE: chunk is the letter which is repeating 2 or more than 2 times.
Include a class **UserMainCode** with a static method **largestChunk** which accepts a string. The return type (Integer) should return the length of the largest chunk if the chunk is present, else return -1.

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

Input and Output Format:

Input consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

You are toooo good

Sample Output 1:

4

(Because the largest chunk is letter 'o' which is repeating 4 times)

Sample Input 2:

who are u

Sample Output 2:

No chunks

```
import java.util.*;

public class Main {

public static void main(String[] args) {

String s1="You are toooo good";

System.out.println(maxChunk(s1));

}

public static int maxChunk(String s1) {

int max=0;

StringTokenizer t=new StringTokenizer(s1," ");

while(t.hasMoreTokens()){

String s2=t.nextToken();

int n=0;

for(int i=0;i<s2.length()-1;i++)

if(s2.charAt(i)==s2.charAt(i+1))

n++;

if(n>max)

max=n;
```

```
}

return (max+1);

}

}
```

44) Unique Characters in a string

Write a program that takes a string and returns the number of unique characters in the string. If the given string doest not contain any unique characters return -1

Include a class **UserMainCode** with a static method **uniqueCounter** which accepts a string as input.

The return type of the output is the count of all unique characters in the strings.

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

Input and Output Format:

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

HelloWorld

Sample Output 1:

5

Sample Input 2:

coco

Sample Output 2:

-1

```
public class Main {
public static void main(String[] args) {
String s1="HelloWorld";
getvalues(s1);
}
public static void getvalues(String s1) {
String s2=s1.toLowerCase();
StringBuffer sb=new StringBuffer(s2);
int l=sb.length();
int count=0;
for(int i=0;i<l;i++)
{ count=0;
for(int j=i+1;j<l;j++)
{
if(sb.charAt(i)==sb.charAt(j))
{
sb.deleteCharAt(j);
count++;
j--;
l--;
j=i;
}
```

```
}
if(count>0)
{
sb.deleteCharAt(i);
i--;
l--;
}
}
if(sb.length()==0)
{
System.out.println(-1);
}
else
System.out.println(sb.length());
}
}
```

45) Name Shrinking

Write a program that accepts a string as input and converts the first two names into dot-separated initials and printa the output.

Input string format is 'fn mn ln'. Output string format is 'ln [mn's 1st character].[fn's 1st character]'

Include a class **UserMainCode** with a static method **getFormatedString** which accepts a string. The return type (String) should return the shrinked name.

Create a Class Main which would be used to accept Input 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:
Sachin Ramesh Tendulkar
Sample Output:
Tendulkar R.S

```
import java.util.Scanner;
public class Main {


	public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
System.out.println(UserMainCode.getFormatedString(s));



}
}
```

```
import java.util.StringTokenizer;


public class UserMainCode {
	public static String getFormatedString(String s)

	{
		StringTokenizer st=new StringTokenizer(s," ");
		StringBuffer sb=new StringBuffer();
		while(st.hasMoreTokens())
		{
		String a=st.nextToken();
		String b=st.nextToken();
		String c=st.nextToken();
		sb.append(c.substring(0));
		sb.append(" ");
		sb.append(b.substring(0,1));
```

```
            sb.append(".");
            sb.append(a.substring(0,1));
            //String ss=sb.toString();
            }
            return sb.toString();
    }
```

46) Odd Digit Sum

Write a program to input a String array. The input may contain digits and alphabets ("de5g4G7R"). Extract odd digits from each string and find the sum and print the output.
For example, if the string is "AKj375A" then take 3+7+5=15 and not as 375 as digit.
Include a class **UserMainCode** with a static method **oddDigitSum** which accepts a string array and the size of the array. The return type (Integer) should return the sum.

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

Assume maximum length of array is 20.

Input and Output Format:

Input consists of an integer n, corresponds to the number of strings, followed by n Strings.

Output consists of an Integer.

Refer sample output for formatting specifications.

Sample Input :
3
cog2nizant1
al33k
d2t4H3r5
Sample Output :
15
(1+3+3+3+5)

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

```
  public class kape {
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int s1=sc.nextInt();
        String[] s2 = new String[s1];
        for (int i = 0; i < s1; i++) {
        s2[i] = sc.next();
        }
System.out.println(kape1.getSum(s2));
}}

public class kape1 {

public static int getSum(String[] s1) {
int sum=0;
for(int i=0;i<s1.length;i++)
for(int j=0;j<s1[i].length();j++){
char c=s1[i].charAt(j);
if(Character.isDigit(c)){
if(c%2!=0)
{
String t=String.valueOf(c);
```

```
int n=Integer.parseInt(t);

sum=sum+n; } }}

return sum;

 }

}
```

47) Unique Number

Write a program that accepts an Integer as input and finds whether the number is Unique or not. Print Unique if the number is “Unique”, else print “Not Unique”.

Note: A Unique number is a positive integer (without leading zeros) with no duplicate digits.For example 7, 135, 214 are all unique numbers whereas 33, 3121, 300 are not.

Include a class **UserMainCode** with a static method **getUnique** which accepts an integer. The return type (Integer) should return 1 if the number is unique else return -1.

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

Input and Output Format:

Input consists of an integer .

Output consists of a String (“Unique” or “Not Unique”).

Refer sample output for formatting specifications.

Sample Input 1:
123
Sample Output 1:
Unique

Sample Input 2:
33
Sample Output 2:
Not Unique

```
public class useer{
```

```
public static void main(String[]args)
{
	Scanner sc=new Scanner(System.in);
	int n=sc.nextInt();
	int []a=new int[100];
	int i=0,count=0;
	while(n!=0)
	{
		int num=n%10;
		a[i]=num;
		i++;
		n=n/10;
	}
	for(int j=0;j<i-1;j++)
	{
		for(int k=j+1;k<=i-1;k++)
		{
			if(a[j]==a[k]){
				count++;
			}
		}}
			if(count>0)
```

```
{
        System.out.println("Invalid/not unique");
}
else
{
        System.out.println("valid/unique");
}
}}
```

48) Sum of Lowest marks

Given input as HashMap, value consists of marks and rollno as key.Find the sum of the lowest three subject marks from the HashMap.

Include a class **UserMainCode** with a static method **getLowest** which accepts a Hashmap with marks and rollno.

The return type of the output is the sum of lowest three subject marks.

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

Input and Output Format:

First line of the input corresponds to the HashMap size.

Input consists a HashMap with marks and rollno.

Output is an integer which is the sum of lowest three subject marks.

Refer sample output for formatting specifications.

Sample Input 1:

```
5
1
54
2
85
3
```

74
4
59
5
57

Sample Output 1:

170

Sample Input 2:

4
10
56
20
58
30
87
40
54

Sample Output 2:

168

49) Color Code Validation

Give a String as colour code as input and write code to validate whether the given string is a valid color code or not.

Validation Rule:
String should start with the Character '#'.
Length of String is 7.
It should contain 6 Characters after '#' Symbol.
It should contain Characters between 'A-F' and Digits '0-9'.
If String acceptable the return true otherwise false.

Include a class **UserMainCode** with a static method **validateColourCode** which accepts a string as input.

The return type of the output is a boolean which returns true if its is a valid color code else it returns false.

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

Input and Output Format:

Input consists a string corresponding to the color code.

Output is a boolean which returns true or false

Refer sample output for formatting specifications.

Sample Input 1:

#99FF33

Sample Output 1:

true

Sample Input 2:

#CCCC99#

Sample Output 2:

False

```
import java.util.Scanner;

class Main
{
	public static void main(String[] a)
	{
		Scanner sc=new Scanner(System.in);
		String s=sc.nextLine();
		if(s.matches("(#)[A-Z0-9]{6}"))
		{
			System.out.println("valid");
		}
		else
			System.out.println("invalid");

	}
}
```

50) Repeating set of characters in a string

Get a string and a positive integer n as input .The last n characters should repeat the number of times given as second input.Write code to repeat the set of character from the given string.

Include a class **UserMainCode** with a static method **getString** which accepts a string and an integer n as input.

The return type of the output is a string with repeated n characters.

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

Input and Output Format:

Input consists a string and a positive integer n.

Output is a string with repeated characters.

Refer sample output for formatting specifications.

Sample Input 1:

Cognizant
3

Sample Output 1:

Cognizantantantant

Sample Input 2:

myacademy
2

Sample Output 2:

Myacademymymy

```
import java.util.*;

public class useerm {

	public static String lengthiestString(String s1,int n){

		StringBuffer sb=new StringBuffer();

		sb.append(s1);

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


		{
```

```
                    sb.append(s1.substring(s1.length()-n,s1.length()));
// sb.append(s1.substring(s1.length()-n))


        }
    return sb.toString();
        }
        public static void main(String[] args) {
                Scanner s=new Scanner(System.in);
                System.out.println("enter the String:");
                String s1=s.nextLine();
                int n=s.nextInt();
                System.out.println("the lengthiest string is:"+lengthiestString(s1,n));
                }
}
```

51) Finding the day of birth

Given an input as date of birth of person, write a program to calculate on which day (MONDAY,TUESDAY....) he was born store and print the day in Upper Case letters.

Include a class **UserMainCode** with a static method **calculateBornDay** which accepts a string as input.

The return type of the output is a string which should be the day in which the person was born.

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

Input and Output Format:

NOTE: date format should be(dd-MM-yyyy)
Input consists a date string.

Output is a string which the day in which the person was born.

Refer sample output for formatting specifications.

Sample Input 1:

29-07-2013
Sample Output 1:

MONDAY

Sample Input 2:

14-12-1992
Sample Output 2:

MONDAY

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Main {
	public static void main(String[] args) throws ParseException {

		Scanner sc=new Scanner(System.in);
	   String s1=sc.nextLine();
	   System.out.println(UserMainCode.calculateBornDay(s1));
		}
		}

import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
public class UserMainCode {
	public static String calculateBornDay(String s1) throws ParseException
	{
		SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
		SimpleDateFormat sdf1=new SimpleDateFormat("EEEEE");
		Date d=sdf.parse(s1);
		String s=sdf1.format(d);
		return s.toUpperCase();
	}
}
```

52) Removing elements from HashMap

Given a HashMap as input, write a program to perform the following operation : If the keys are divisible by 3 then remove that key and its values and print the number of remaining keys in the hashmap.

Include a class **UserMainCode** with a static method **afterDelete** which accepts a HashMap as input.

The return type of the output is an integer which represents the count of remaining elements in the hashmap.

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

Input and Output Format:

First input corresponds to the size of hashmap

Input consists a HashMap

Output is an integer which is the count of remaining elements in the hashmap.

Refer sample output for formatting specifications.

Sample Input 1:

4
339
RON
1010
JONS
3366
SMITH
2020
TIM

Sample Output 1:

2

Sample Input 2:

5
1010
C2WE

6252
XY4E
1212
M2ED
7070
S2M41ITH
8585
J410N

Sample Output 2:

3

53) Experience Calculator

Write a program to read Date of Joining and current date as Strings and Experience as integer and validate whether the given experience and calculated experience are the same. Print “true” if same, else “false”.

Include a class **UserMainCode** with a static method **calculateExperience** which accepts 2 strings and an integer. The return type is boolean.

Create a Class Main which would be used to accept 2 string (dates) and an integer and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 2 strings and an integer, where the 2 strings corresponds to the date of joining and current date, and the integer is the experience.

Output is either “true” or “false”.

Refer sample output for formatting specifications.

Sample Input 1:
11/01/2010
01/09/2014
4
Sample Output 1:
true

Sample Input 2:
11/06/2009
01/09/2014
4
Sample Output 2:
False

```
import java.util.Date;
import java.text.SimpleDateFormat;
public class Usermaincode
{public static boolean display(String s,String s1,int n)
{
boolean b=false;
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
try{
Date d=sdf.parse(s);
Date d1=sdf.parse(s1);
int y=d.getYear();
int y1=d1.getYear();
int m=d.getMonth();
int m1=d1.getMonth();
int day=d.getDay();
int day1=d1.getDay();
int age=y1-y;
if(m>m1)
age--;
else if(m==m1)
{if(day<day1)
```

```
age--;

}

if(age==n)

b=true;

else

b=false;

}

catch(Exception e)

{e.printStackTrace();

}

return b;

}

}
```

54) Flush Characters

Write a program to read a string from the user and remove all the alphabets and spaces from the String, and **only store special characters and digit** in the output String. Print the output string.

Include a class **UserMainCode** with a static method **getSpecialChar** which accepts a string. The return type (String) should return the character removed string.

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

Input and Output Format:

Input consists of a strings.

Output consists of an String (character removed string).

Refer sample output for formatting specifications.

Sample Input :
cogniz$#45Ant
Sample Output :
$#45

```
public class User {
      public static String repeatString (String s)
      {
            StringBuffer sb=new StringBuffer();
            for(int i=0;i<s.length();i++)
            {

                              /* char c=s.charAt(i);
                        if(!Character.isAlphabetic(c)) */


            // if(!Character.isAlphabetic(s.charAt(i))
&& (!charachter.isWhiteSpace(s.CharAt(i)))

if( (!Character.isAlphabetic(s.charAt(i)))
                              &&  (!Character.isWhitespace(s.charAt(i))) )

                  sb.append(s.charAt(i));
            }
            return sb.toString();
       }
}
```

55) String Repetition

Write a program to read a string and an integer and return a string based on the below rules.

If input2 is equal or greater than 3 then repeat the first three character of the String by given input2 times, separated by a space.

If input2 is 2 then repeat the first two character of String two times separated by a space,

If input2 is 1 then return the first character of the String.

Include a class UserMainCode with a static method **repeatString** which takes a string & integer and returns a string based on the above rules.

Create a Class Main which would be used to accept Input string 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:

COGNIZANT

4

Sample Output 1:

COG COG COG COG

Sample Input 2:

COGNIZANT

2

Sample Output 2:

CO CO

```
public class User {
	public static String repeatString (String s,int n)
	{
		StringBuffer sb=new StringBuffer();
		if(n>=3)
		{
			for(int i=0;i<n;i++)
			{
			sb.append(s.substring(0,3)).append(" ");
			}
		}
		else if(n==2)
		{
			for(int i=0;i<n;i++)
			sb.append(s.substring(0,2)).append(" ");
		}
		else if(n==1)
		{
			for(int i=0;i<n;i++)
			sb.append(s.substring(0,1)).append(" ");
		}

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

56) Average of Prime Locations

Write a program to read an integer array and find the average of the numbers located on the Prime location(indexes).

Round the avarage to two decimal places.

Assume that the array starts with index 0.

Include a class UserMainCode with a static method **averageElements** which accepts a single integer array. The return type (double) should be the average.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Double value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

8

4

1

7

6

5

8

6

9

Sample Output 1:

7.5

```
public class User {
	public static float averageElements(int a[],int n)
	{
		int c=0,sum=0,k=0;
		float avg=0;
		for(int i=2;i<=n;i++)
		{
			c=0;
			for(int j=1;j<i;j++)
			{
			if(i%j==0)
				c++;

		}
			if(c==1)
			{
				k++;
				sum=sum+a[i];
			}
		}
		avg=(float)sum/k;
return avg;
	 }
}
```

57) Common Elements

Write a program to read two integer arrays and find the sum of common elements in both the arrays. If there are no common elements return -1 as output

Include a class UserMainCode with a static method **sumCommonElements** which accepts two single integer array. The return type (integer) should be the sum of common elements.

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

Assume that all the elements will be distinct.

Input and Output Format:

Input consists of 2n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array, The last n elements correspond to the elements of the second array.

Output consists of a single Integer value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

4

1

2

3

4

2

3

6

7

Sample Output 1:

5

```
public class User {
	public static int getMiddleElement (int a[],int b[],int n)
	{
		int sum=0;
		for(int i=0;i<n;i++)
		{
			for(int j=0;j<n;j++)
			{
			if(a[i]==b[j])
				sum=sum+a[i];

		}
		}
return sum;
	 }
}
```

58) Middle of Array

Write a program to read an integer array and return the middle element in the array. The size of the array would always be odd.

Include a class UserMainCode with a static method **getMiddleElement** which accepts a single integer array. The return type (integer) should be the middle element in the array.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 19.

Sample Input 1:

5

1

5

23

64

9

Sample Output 1:

23

```
public class User {
	public static int getMiddleElement (int a[])
	{
		int n=a.length;
		int mid=n/2;
		return a[mid];

	 }
}
```

59) Simple String Manipulation

Write a program to read a string and return a modified string based on the following rules.

Return the String without the first 2 chars except when

1. keep the first char if it is 'j'
2. keep the second char if it is 'b'.

Include a class UserMainCode with a static method **getString** which accepts a string. The return type (string) should be the modified string based on the above rules. Consider all letters in the input to be small case.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

hello

Sample Output 1:

llo

Sample Input 2:

java

Sample Output 2:

Jva

```
public class User {
	public static String getString(String s)
	{
		StringBuffer sb=new StringBuffer();
		char a=s.charAt(0);
		char b=s.charAt(1);
		if(a!='j'&& b!='b')
			sb.append(s.substring(2));
			else if(a=='j' && b!='b')
			sb.append("j").append(s.substring(2));
			else if(a!='j' && b=='b')
			sb.append(s.substring(1));
			else
			sb.append(s.substring(0));
			return sb.toString();
```

```
        }
}
```

60) Date Validation

Write a program to read a string representing a date. The date can be in any of the three formats

1:dd-MM-yyyy 2: dd/MM/yyyy 3: dd.MM.yyyy

If the date is valid, print **valid** else print **invalid**.

Include a class UserMainCode with a static method **getValidDate** which accepts a string. The return type (integer) should be based on the validity of the date.

Create a Class Main which would be used to accept Input 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:

03.12.2013

Sample Output 1:

valid

Sample Input 2:
03$12$2013
Sample Output 3:
Invalid

```
public class User {
      public static int getValidDate(String s) throws ParseException
      {
            int res=0;
            // String s1=null,s2=null,s3=null;
            if(s.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))
            {
         SimpleDateFormat sdf1=new SimpleDateFormat("dd.MM.yyyy");
         sdf1.setLenient(false);
         try
         {
```

```
          Date d1=sdf1.parse(s);
          res=1;
          }
          catch (ParseException e)
          {
                res=-1;
          }
          }

             else if(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-]{1}[0-9]{4}"))
             {

                              SimpleDateFormat sdf3=new SimpleDateFormat("dd-MM-
yyyy");
                              sdf3.setLenient(false);
                              try
                              {
                              Date d1=sdf3.parse(s);
                              res=1;
                              }
                              catch (ParseException e)
                              {
                                    res=-1;
                              }
              }
             else if(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))
             {

                              SimpleDateFormat sdf3=new
SimpleDateFormat("dd/MM/yyyy");
                              sdf3.setLenient(false);
                              try
                              {
                              Date d1=sdf3.parse(s);
                              res=1;
                              }
                              catch (ParseException e)
                              {
                                    res=-1;
                              }
              }

             else
                res=0;

          return res;
       }
}
```

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;


	}
}
```

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

```
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;
      }
}
```

81) String Processing - V

Write a program to read a string array, concatenate the array elements one by one separated by comma and return the final string as output.
Include a class UserMainCode with a static method **concatString** which accepts the string array. The return type is the string.
Create a Class Main which would be used to accept the string array and call the static method present in UserMainCode.

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n string values.
Output consists of the string.
Refer sample output for formatting specifications.

Sample Input 1:
3

AAA
BBB
CCC
Sample Output 1:
AAA,BBB,CCC

```
public class Main {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		String[] s=new String[n];
		for(int i=0;i<n;i++)
			s[i]=sc.next();
		System.out.println(User.concatString(s));
	}
}
```

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

82) Unique Number

GIven three integers (a,b,c) , Write a program that returns the number of unique integers among the three.
Include a class UserMainCode with a static method **calculateUnique** which accepts three integers and returns the count as 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:
12

4
3

Sample Output 1:

3

Sample Input 2:

4
-4
4

Sample Output 2:

2

```
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();
		System.out.println(User.calculateUnique(a,b,c));
	}
}

public class User {
	public static int calculateUnique(int a,int b,int c)
	  {
    int count=0;
    int[] s={a,b,c};
    int[] res=new int[3];
    for(int i=0;i<s.length;i++)
    {
        res[i]=Math.abs(s[i]);
    }
    count=0;
    for(int i=0;i<res.length-1;i++)
    {

      if(res[i]==res[i+1])
      {
            count++;
      }
    }
    return count+1;
      }
}



public class Main {

	public static void main(String[] args) {
	 int ct1=0;
		Scanner sc=new Scanner(System.in);
```

```
            int a=sc.nextInt();
            int b=sc.nextInt();
            int c=sc.nextInt();

                if(a!=b)
                  ct1=ct1+1;

                if(a!=c)
                ct1=ct1+1;

                if(b!=c)
                ct1=ct1+1;

      if((a==b) & (b==c))
            System.out.println("output "+(ct1+1));
      else

            System.out.println("output "+ct1);

      }
}




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 d=0;
            if(a!=b&&a!=c)
            {
                  d=3;
            }
            else if(a==b&&a==c)
            {
                  d=1;
            }
            else if(a!=b&&a==c)

            {
                  d=2;
            }
            else if(a==b&&a!=c)
            {
                  d=2;
            }
```

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

83) Math Calculator

Write a program that accepts three inputs, first two inputs are operands in int form and third one being one of the following five operators: +, -, *, /, %. Implement calculator logic and return the result of the given inputs as per the operator provided. In case of division, Assume the result would be integer.
Include a class UserMainCode with a static method **calculator** which accepts two integers, one operand and returns the 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 two integers and a character.
Output consists of a integer.
Refer sample output for formatting specifications.

Sample Input 1:
23
2
*
Sample Output 1:
46

```
public class Main {
      public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            int a=sc.nextInt();
            int b=sc.nextInt();
            char op=sc.next().charAt(0);
            System.out.println(User.calculateUnique(a,b,op));
      }
}
```

```
public class User {
      public static int calculateUnique(int a,int b,char op)
      {
    int res=0;
    switch(op){

    case '+':
```

```
      res=a+b;
    case '-':
      res=Math.abs(a-b);
    case '*':
      res=a*b;
    case '/':
      res=Math.round(a/b);
    case '%':
      res=Math.round(a%b);
    }
    return res;
      }
}
```

84) Scores

Write a program to read a integer array of scores, if 100 appears at two consecutive locations return true else return false.
Include a class UserMainCode with a static method **checkScores** which accepts the integer array. The return type is boolean.
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of a string that is either 'TRUE' or 'FALSE'.
Refer sample output for formatting specifications.

Sample Input 1:
3
1
100
100
Sample Output 1:
TRUE

Sample Input 2:
3
100
1
100
Sample Output 2:
FALSE

```
public class User {
      public static boolean scanArray(int s[])
      {
   boolean b=false;

         for(int i=0;i<s.length-1;i++)
         {
               if(s[i]==100&&s[i+1]==100)
               {
                     b=true;
               break;
               }
               else
                     b=false;
         }


   return b;
      }
}
```

85) ArrayFront

Write a program to read a integer array and return true if one of the first 4 elements in the array is 9 else return false.
Note: The array length may be less than 4.
Include a class UserMainCode with a static method **scanArray** which accepts the integer array. The return type is true / false.
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of TRUE / FALSE.
Refer sample output for formatting specifications.

Sample Input 1:
6
1
2
3
4
5

6

Sample Output 1:

FALSE

Sample Input 2:

3
1
2
9

Sample Output 2:

TRUE

```
public class Main {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		int[] s= new int[n];
		for(int i=0;i<n;i++)
			s[i]=sc.nextInt();
		boolean b=User.scanArray (s);
		System.out.println(b);
	}
}
```

```
public class User {
	public static boolean scanArray(int s[])
	{
   boolean b=false;
   if(s.length>4)
   {
		for(int i=0;i<4;i++)
		{
			if(s[i]==9)
				b=true;
			else
				b=false;
		}
   }
   else
   {
		for(int i=0;i<s.length;i++)
		{
			if(s[i]==9)
				b=true;
			else
				b=false;
		}
   }
   return b;
	}
```

}

86) Word Count

Given a string array (s) and non negative integer (n) and return the number of elements in the array which have same number of characters as the givent int N.
Include a class UserMainCode with a static method **countWord** which accepts the string array and integer. The return type is the string formed based on rules.
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 an integer indicating the number of elements in the string array followed the elements and ended by the non-negative integer (N).
Output consists of a integer .
Refer sample output for formatting specifications.

Sample Input 1:
4
a
bb
b
ccc
1
Sample Output 1:
2

Sample Input 2:
5
dog
cat
monkey
bear
fox
3
Sample Output 2:
3

```
public class Main {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
```

```
            String[] s= new String[n];
            for(int i=0;i<n;i++)
                  s[i]=sc.next();
            int n1=sc.nextInt();
            System.out.println(User.countWord (s,n1));
      }
}
```

```
public class User {
      public static int countWord (String s[],int n1)
      {
   int count=0;
   for(int i=0;i<s.length;i++)
   {
         if(s[i].length()==n1)
               count++;
   }
   return count;
      }
}
```

87) Find Distance

Write a Program that accepts four int inputs(x1,y1,x2,y2) as the coordinates of two points. Calculate the distance between the two points using the below formula.
Formula : square root of((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))
Then, Round the result to return an int
Include a class UserMainCode with a static method **findDistance** which accepts four integers. The return type is integer representing the formula.
Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

Input and Output Format:
Input consists of four integers.
Output consists of a single integer.
Refer sample output for formatting specifications.

Sample Input 1:
3
4
5
2
Sample Output 1:
3

Sample Input 2:
3

1
5
2
Sample Output 2:
2

```
public class Main {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		int x1=sc.nextInt();
		int y1=sc.nextInt();
		int x2=sc.nextInt();
		int y2=sc.nextInt();
		System.out.println(User.findDistance(x1,y1,x2,y2));
	}
}
```

```
public class User {
	public static int findDistance(int x1,int y1,int x2,int y2)
	{
    double d=((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
	  int res=(int)Math.ceil(Math.sqrt(d));
	  return res;
	  }
}
```

88) Word Count - II

Write a program to read a string and count the number of words present in it.
Include a class UserMainCode with a static method **countWord** which accepts the string. The return type is the integer giving out the count of words.
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 integer.
Refer sample output for formatting specifications.

Sample Input 1:
Today is Sunday
Sample Output 1:
3

```
public class Main {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
```

```
            String s=sc.nextLine();
            System.out.println(User.countWord(s));
      }
}
```

```
public class User {
      public static int countWord(String s)
      {
      StringTokenizer st=new StringTokenizer(s," ");
      int count =st.countTokens();
      return count;
      }
}
```

89) Sum of Max & Min

Write a Program that accepts three integers, and returns the sum of maximum and minimum numbers.
Include a class UserMainCode with a static method getSumMaxMin which accepts three integers. The return type is integer representing the formula.
Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

Input and Output Format:
Input consists of three integers.
Output consists of a single integer.
Refer sample output for formatting specifications.

Sample Input 1:
12
17
19

Sample Output 1:
31

```
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();
            System.out.println(User.getSumMaxMin (a,b,c));
      }
}
```

```
public class User {
	public static int getSumMaxMin (int a,int b,int c)
	{
	int sum=0;
	int[] s={a,b,c};
	Arrays.sort(s);
	sum=s[0]+s[2];
	return sum;

	}
}
```

90) Decimal to Binary Conversion

Write a Program that accepts a decimal number n, and converts the number to binary.

Include a class UserMainCode with a static method **convertDecimalToBinary** which accepts an integer. The return type is long representing the binary number.

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

Input and Output Format:
Input consists of single integer.
Output consists of a single long.
Refer sample output for formatting specifications.

Sample Input 1:
5
Sample Output 1:
101

```
public class Main {
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		System.out.println(User.convertDecimalToBinary(n));
	}
}
```

```
public class User {
	public static long convertDecimalToBinary(int n)
	{
	String x=   Integer.toBinaryString(n);
	long res= Integer.parseInt(x);
```

```
            long res=Integer.parseInt(Integer.toBinaryString(n));


        return res;
        }
}
```

91) String Processing - V

Write a program to read a string and also a number N. Form a new string made up of n repetitions of the last n characters of the String. You may assume that n is between 1 and the length of the string.

Include a class UserMainCode with a static method **returnLastRepeatedCharacters** 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:
Hello
2
Sample Output 1:
lolo

Sample Input 2:
Hello
3
Sample Output 2:
llollollo

```
public class Main {
      public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            String s1=sc.next();
            int n=sc.nextInt();
            System.out.println(User.returnLastRepeatedCharacters (s1,n));
      }
}

public class User {
      public static String returnLastRepeatedCharacters (String s1,int n)
      {
            StringBuffer sb=new StringBuffer();
            for(int i=0;i<n;i++)
```

```
            sb.append(s1.substring(s1.length()-n));
             return sb.toString();
      }
}
```

92) Regular Expression - III

Given a string (s) apply the following rules.
1. String should not begin with a number.
If the condition is 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:
ab2
Sample Output 1:
TRUE

Sample Input 2:
72CAB
Sample Output 2:
FALSE

```
public class Main {
      public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            String s1=sc.next();
            boolean b=User.validateString(s1);
            System.out.println(b);
      }
}
```

```
public class User {
      public static boolean validateString(String s1)
      {
            boolean b= false;
            if(!Character.isDigit(s1.charAt(0)))
                  b= true;
            else
                  b= false;
```

```
            return b;

      }
}
```

93) 3String Processing - TrimCat

Write a program to read a string and return a new string which is made of every alternate characters starting with the first character. For example NewYork will generate Nwok, and Samurai will generate Smri.

Include a class UserMainCode with a static method getAlternateChars 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:
Hello

Sample Output 1:
Hlo

```
public class Main {


      public static void main(String[] args) {
            Scanner sc=new Scanner(System.in);
            String s1=sc.next();
            System.out.println(User.fetchUserName (s1));
      }
}

public class User {
      public static String fetchUserName (String s1)
      {
            StringBuffer sb=new StringBuffer();
            for( int i=0;i<s1.length();i=i+2)
                  sb.append(s1.charAt(i));
            return sb.toString();
```

```
	}
}
```

94) 2 String Processing - Username

Write a program to read a valid email id and extract the username.

Note - user name is the string appearing before @ symbol.

Include a class UserMainCode with a static method fetchUserName 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:
admin@xyz.com

Sample Output 1:
admin

```
public class Main {


	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		String s1=sc.next();
		System.out.println(User.fetchUserName (s1));
	}
}
public class User {
	public static String fetchUserName (String s1)
	{
		boolean b=false;
		StringTokenizer st=new StringTokenizer(s1,"@");
		String name=st.nextToken();

		return name;

	}
}
```

95) 1 String Processing - VII

Write a program to read a two strings and one int value(N). check if Nth character of first String from start and Nth character of second String from end are same or not. If both are same return true else return false.
Check need not be Case sensitive
Include a class UserMainCode with a static method **isEqual** which accepts the two strings and a integer n. The return type is the TRUE / FALSE.
Create a Class Main which would be used to read the strings and integer and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two strings and an integer.
Output consists of TRUE / FALSE .
Refer sample output for formatting specifications.

Sample Input 1:
AAAA
abab
2

Sample Output 1:
TRUE

Sample Input 2:
MNOP
QRST
3

Sample Output 2:
FALSE

```
public class Main {


	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
		String s1=sc.next();
		String s2=sc.next();
		int n=sc.nextInt();
		boolean b=User.isEqual(s1,s2,n);
		System.out.println(b);
```

```
	}

}
```

```
public class User {
	public static boolean isEqual(String s1,String s2, int n)
	{
		boolean b=false;
		String i= s1.toLowerCase();
		String j= s2.toLowerCase();
		if(i.charAt(n-1)==j.charAt(n))
			b=true;
		else
			b=false;

		return b;

	}
}
```

96) Largest Difference

Write a program to read a integer array, find the largest difference between adjacent elements and display the index of largest difference.
EXAMPLE:
input1: {2,4,5,1,9,3,8}
output1: 4 (here largest difference 9-1=8 then return index of 9 ie,4)
Include a class UserMainCode with a static method **checkDifference** which accepts the integer array. The return type is integer.

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

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of integer.
Refer sample output for formatting specifications.

Sample Input 1:
7
2
4
5
1
9
3
8

Sample Output 1:

4

```
public class Main {
	public static int getDiffArray(int[] n1){
		int n2,n3=0,n4=0,i;
		for(i=0;i<n1.length-1;i++){
			n2=Math.abs(n1[i]-n1[i+1]);
			if(n2>n3){
				n3=n2;
				n4=i+1; }}
		return n4;
	}
	public static void main(String[] args) {
		int[] n1={2,4,5,1,9,3,8};
		System.out.println(getDiffArray(n1));
	}
}
```

1. **Start Case**

Write a program to read a sentence in string variable and convert the first letter of each word to capital case. Print the final string.
Note: - Only the first letter in each word should be in capital case in final string.
Include a class **UserMainCode** with a static method **printCapitalized** which accepts a string. The return type (String) should return the capitalized string.

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

Input and Output Format:

Input consists of a strings.

Output consists of a String (capitalized string).

Refer sample output for formatting specifications.

Sample Input:
Now is the time to act!
Sample Output:
Now Is The Time To Act!

Solution :

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s = sc.nextLine();
System.out.println(User.printCapitalized(s));
}
}
```

```
import java.util.StringTokenizer;

publicclass User {
publicstatic String printCapitalized(String s) {
StringTokenizer st= new StringTokenizer(s," ");
StringBuffer sb= new StringBuffer();
while(st.hasMoreTokens())
{
      String s1=st.nextToken();
      String s2=s1.substring(0, 1);
      String s3=s1.substring(1);
      sb.append(s2.toUpperCase());
      sb.append(s3);
      sb.append(" ");
```

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

```
import java.util.StringTokenizer;
publicclass UserMainCode {
        publicstatic String changeWord(String s)
        {
                StringTokenizer st=new StringTokenizer(s," ");
                StringBuffer sb=new StringBuffer();
                while(st.hasMoreTokens())
                {
                        String s1=st.nextToken();
                        sb.append(s1.substring(0,1).toUpperCase());
                        sb.append(s1.substring(1));
                        sb.append(" ");

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

2. **Maximum Difference**

Write a program to read an integer array and find the index of larger number of the two adjacent numbers with largest difference. Print the index.

Include a class **UserMainCode** with a static method **findMaxDistance** which accepts an integer array and the number of elements in the array. The return type (Integer) should return index.

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

Input and Output Format:

Input consists of n+1 integers, where n corresponds the size of the array followed by n integers.

Output consists of an Integer (index).

Refer sample output for formatting specifications.

Sample Input :

6
4
8
6
1
9
4

Sample Output :

4

[In the sequence 4 8 6 1 9 4 the maximum distance is 8 (between 1 and 9). The function should return the index of the greatest of two. In this case it is 9 (which is at index 4). output = 4.]

Solution :

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
 a[i]=sc.nextInt();

System.out.println(User.findMaxDistance(a));
}
}



publicclass User {
publicstaticint findMaxDistance(int[] a){

      int dif,max=0;
      int n=a.length;
      for(int i=0;i<n-1;i++)
      {
      dif=Math.abs(a[i]-a[i+1]);


      // if(max<dif)

if(dif>max)
      {
            if(a[i+1]>a[i])
            max=i+1;
            else
            max=i;
```

```
}

        }
        return max;
}
}
```

3. Palindrome - In Range

Write a program to input two integers, which corresponds to the lower limit and upper limit respectively, and find the sum of all palindrome numbers present in the range including the two numbers. Print the sum.

Include a class **UserMainCode** with a static method **addPalindromes** which accepts two integers. The return type (Integer) should return the sum if the palindromes are present, else return 0.

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

Note1 : A palindrome number is a number which remains same after reversing its digits.
Note2 : A single digit number is not considered as palindrome.

Input and Output Format:

Input consists of 2 integers, which corresponds to the lower limit and upper limit respectively.

Output consists of an Integer (sum of palindromes).

Refer sample output for formatting specifications.

Sample Input :
130
150
Sample Output :
272
(131+141 = 272)

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
intb = sc.nextInt();
```

```
System.out.println(User.addPalindromes(a,b));
}
}

publicclass User {
publicstaticint addPalindromes(int a, int b){

      int temp=0,sum=0,r,sum1=0;


      for(int i=a;i<=b;i++)
      {
          temp=i;
        sum=0;
while(temp>0){
             r=temp%10;
             sum=sum*10+r;
             temp=temp/10;
         }
if(i==sum)
          sum1=sum1+i;
   }
      return sum1;
}
}
```

4. **PAN Card**

Write a program to read a string and validate PAN no. against following rules:

1. There must be eight characters.
2. First three letters must be alphabets followed by four digit number and ends with alphabet
3. All alphabets should be in capital case.

Print “Valid” if the PAN no. is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **validatePAN** which accepts a string. The return type (Integer) should return 1 if the string is a valid PAN no. else return 2.

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

Input and Output Format:

Input consists of a string, which corresponds to the PAN number.

Output consists of a string - "Valid" or "Invalid"

Refer sample output for formatting specifications.

Sample Input 1:

ALD3245E

Sample Output 1:

Valid

Sample Input 2:

OLE124F

Sample Output 2:

Invalid

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s=sc.next();
int res=User.validatePAN(s);
if(res==1)
	System.out.println("Valid");
else
	System.out.println("Invalid");

}
}


publicclass User {
publicstaticint validatePAN(String s){

	int res=0;
	if(s.length()==8)
	{
	if(s.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))
		res=1;
	else
		res=2;
	}
	return res;
}
}
```

5. **Fibonacci Sum**

Write a program to read an integer n, generate fibonacci series and calculate the sum of first n numbers in the series. Print the sum.

Include a class **UserMainCode** with a static method **getSumOfNfibos** which accepts an integer n. The return type (Integer) should return the sum of n fibonacci numbers.

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

Note: First two numbers in a Fibonacci series are 0, 1 and all other subsequent numbers are sum of its previous two numbers. Example - 0, 1, 1, 2, 3, 5...

Input and Output Format:

Input consists of an integer, which corresponds to n.

Output consists of an Integer (sum of fibonacci numbers).

Refer sample output for formatting specifications.

Sample Input :

5

Sample Output :

7

[0 + 1 + 1 + 2 + 3 = 7]

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int s=sc.nextInt();

      System.out.println(User.getSumOfNfibos(s));


}
}

publicclass User {
publicstaticint getSumOfNfibos(int s){


      int a=0,b=1,c=0,d=1;
      for(int i=3;i<=s;i++)
      {
            c=a+b;
            a=b;
            b=c;
            d=d+c;

      }
      returnd;
}
}
```

6. **Test Vowels**

Write a program to read a string and check if given string contains exactly five vowels in any order. Print “Yes” if the condition satisfies, else print “No”.
Assume there is no repetition of any vowel in the given string and all characters are lowercase.
Include a class **UserMainCode** with a static method **testVowels** which accepts a string. The return type (Integer) should return 1 if all vowels are present, else return 2.

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

Input and Output Format:

Input consists of a string.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

Sample Input 1:
acbisouzze
Sample Output 1:
Yes

Sample Input 2:
cbisouzze
Sample Output 2:
No

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s=sc.next();
int res=User.testVowels (s);
if(res==1)
      System.out.println("Yes");
else
      System.out.println("No");
}
}

publicclass User {
publicstaticint testVowels (String s){

      int res,count=0;
      String s1="aeiou";
```

```
        String s2=s.toLowerCase();
        for(int i=0;i<s2.length();i++)
        {
                for(int j=0;j<s1.length();j++)
                {
        if(s2.charAt(i)==s1.charAt(j))
        {
                count++;

        }
        }
        }

        if(count==s1.length())
                res=1;
        else
                res=2;
        return res;
}
}
```

```
publicclass User {
publicstaticint testOrderVowels(String s1) {

StringBuffer sb = newStringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}
```

7 . Dash Check

Write a program to read two strings and check whether or not they have dashes in the same places. Print “Yes” if the condition satisfies, else print “No”.
Include a class **UserMainCode** with a static method **compareDashes** which accepts two strings. The return type (Integer) should return 1 if all dashes are placed correctly, else return 2.

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

Note: The strings must have exactly the same number of dashes in exactly the same positions. The strings might be of different length.
Input and Output Format:

Input consists of two strings.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

Sample Input 1:
hi—there-you.
12--(134)-7539
Sample Output 1:
Yes

Sample Input 2:
-15-389
-xyw-zzy
Sample Output 2:
No

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
int res=User.compareDashes (s1,s2);
if(res==1)
      System.out.println("Yes");
else
      System.out.println("No");
}
}

import java.util.ArrayList;
```

```
publicclass User {
publicstaticint compareDashes (String s1,String s2){

int res=0;
      ArrayList<Integer> a1=new ArrayList<Integer>();
      ArrayList<Integer> a2=new ArrayList<Integer>();
      for(int i=0;i<s1.length();i++)
      {
            if(s1.charAt(i)=='-')
                  a1.add(i);
      }
      for(int i=0;i<s2.length();i++)
      {
            if(s2.charAt(i)=='-')
                  a2.add(i);
      }
      if(a1.equals(a2))
            res=1;
      else
            res=2;
      return res;

}
}
```

8. **Reverse Split**

Write a program to read a string and a character, and reverse the string and convert it in a format such that each character is separated by the given character. Print the final string.
Include a class **UserMainCode** with a static method **reshape** which accepts a string and a character. The return type (String) should return the final string.

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

Input and Output Format:

Input consists of a string and a character.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

Sample Input:
Rabbit
-
Sample Output:

t-i-b-b-a-R

```
importjava.util.HashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
System.out.println(User.extractMax(s1,s2));

}
}
```

```
publicclass UserMain {
      publicstaticString extractMax(String s1,String s2){
            StringBuffer sb=new StringBuffer();
            for(int i=0;i<s1.length()-1;i++)
            {
                  sb.append(s1.charAt(i));
                  sb.append(s2);

            }
            sb.append(s1.charAt(s1.length()-1));
            return sb.reverse().toString();
            }
}
```

9. Remove 10's

Write a program to read an integer array and remove all 10s from the array, shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array.

Include a class **UserMainCode** with a static method **removeTens** which accepts the number of elements and an integer array. The return type (Integer array) should return the final array. Create a Class Main which would be used to read the number of elements and the input array, and call the static method present in UserMainCode.

Input and Output Format:

Input consists of n+1 integers, where n corresponds to size of the array followed by n elements of the array.

Output consists of an integer array (the final array).

Refer sample output for formatting specifications.

Sample Input :

5
1
10
20
10
2

Sample Output :

1
20
2
o
o

Solution :

```
 import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
      a[i]=sc.nextInt();
User.removeTens(a);

}
}


importjava.util.ArrayList;


publicclass User {
publicstaticint[] removeTens(int[] a){
      int[] out=newint[a.length];
      int k=0;
for(int i=0;i<a.length;i++)
{
      if(a[i]!=10)
      {
            out[k]=a[i];
            k++;
}}
```

```
for(int i=0;i<a.length;i++)
      System.out.println(out[i]);
return out;
}
}
```

10. Last Letters

Write a program to read a sentence as a string and store only the last letter of each word of the sentence in capital letters separated by $. Print the final string.

Include a class **UserMainCode** with a static method **getLastLetter** which accepts a string. The return type (string) should return the final string.
Create a Class Main which would be used to read a string, and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

Smaple Input :
This is a cat
Sample Output :
SSA$T

Solution :

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

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s= sc.nextLine();
System.out.println(User.getLastLetter(s));

}
}

import java.util.StringTokenizer;


publicclass User {
```

```
publicstatic String getLastLetter(String s){

	StringTokenizer st= new StringTokenizer(s," ");
	String s2=st.nextToken();
	StringBuffer sb= new StringBuffer();
	String s3=Character.toUpperCase(s2.charAt(s2.length()-1))+"";
	while(st.hasMoreTokens())
	{
	 s2=st.nextToken();
	      s3=s3+"$"+Character.toUpperCase(s2.charAt(s2.length()-1));

	}
	returns3;
}
}
```

```
**************************************************************************
******************
```

```
publicclass UserMain {
	publicstatic String getLastLetter(String s)
	{
	StringTokenizer st= new StringTokenizer(s," ");

	StringBuffer sb= new StringBuffer();

	String b=st.nextToken();
	sb.append(b.charAt(b.length()-1));

	while(st.hasMoreTokens())
	{
	String a=st.nextToken();
	sb.append("$");
	sb.append(a.charAt(a.length()-1));

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

}
}
```

```
**************************************************************************
***********
```

11 Largest Key in HashMap

Write a program that construts a hashmap and returns the value corresponding to the largest key.

Include a class UserMainCode with a static method **getMaxKeyValue** which accepts a string. The return type (String) should be the value corresponding to the largest key.

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

Input and Output Format:

Input consists of 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers equals the integer key and value as string.

Output consists of a string which is the value of largest key.

Refer sample output for formatting specifications.

Sample Input 1:

3

12

amron

9

Exide

7

SF

Sample Output 1:

amron

Solutions:

```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n= sc.nextInt();
HashMap<Integer,String> hm= new HashMap<Integer,String>();
for(int i=0;i<n;i++)
        hm.put(sc.nextInt(), sc.next());
System.out.println(User.getMaxKeyValue(hm));
```

```
}
}
```

```
importjava.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstaticString getMaxKeyValue(HashMap<Integer,String> hm){
	int max=0;
	String nn=null;
	Iterator<Integer> it = hm.keySet().iterator();
	while(it.hasNext())
	{
		int key=it.next();
		String name=hm.get(key);
		if(key>max)
		{
			key=max;
		nn=name;
		}
	}
	return nn;
	}
}
```

12. All Numbers

Write a program to read a string array and return 1 if all the elements of the array are numbers, else return -1.

Include a class UserMainCode with a static method **validateNumber** which accepts a string aray. The return type (integer) should be -1 or 1 based on the above rules.

Create a Class Main which would be used to accept Input string array and call the static method present in UserMainCode.
The string array is said to be valid if all the elements in the array are numbers. Else it is invalid.

Input and Output Format:

Input consists of an integer specifying the size of string array followed by n strings.

Refer sample output for formatting specifications.

Sample Input 1:

4
123

24.5

23

one

Sample Output 1:

invalid

Sample Input 2:

2
123

24.5

Sample Output 2:

valid

```
import java.util.HashMap;

import java.util.Scanner;



public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n= sc.nextInt();

String[] s= new String[n];

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

        s[i]=sc.next();

int res=User.validateNumber(s);
```

if(res==1)

System.out.println("Valid");

else

System.out.println("invalid");

}

}

```
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstaticint validateNumber(String s[]){
	int res=0;
	int count=0,temp=0;
	String s1=null;
	for(int i=0;i<s.length;i++)
	{
		s1=s[i];
		count=0;
		for(int j=0;j<s1.length();j++)
		{

		if(s1.charAt(j)>='0'&& s1.charAt(j)<='9' || s1.charAt(j)=='.')
			count++;
		}
	if(count==s1.length())
		temp++;
	}
	if(temp==s.length)
		res=1;
	else
		res=-1;
	return res;
	}
}
```

13. Day of the Week

Write a program to read a date as string (MM-dd-yyyy) and return the day of week on that date.

Include a class UserMainCode with a static method **getDay** which accepts the string. The return type (string) should be the day of the week.

Create a Class Main which would be used to accept Input 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:

07-13-2012

Sample Output 1:

Friday

Solutions :

User :

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;


public class User
{
public static String calculateBornDay(String d) throws ParseException
{
SimpleDateFormat sdf= new SimpleDateFormat("MM-dd-yyyy");
SimpleDateFormat s= new SimpleDateFormat("EEEE");
Date d1= new Date();
```

```
d1= sdf.parse(d);

String day=s.format(d1);

return day;

}

}
```

14. Max Substring

Write a program to accept two string inputs. The first being a source string and second one a delimiter. The source string contains the delimiter at various locations. Your job is to return the substring with maximum number of characters. If two or more substrings have maximim number of characters return the substring which appears first. The size of the delimiter is 1.

Include a class UserMainCode with a static method **extractMax** which accepts the string. The return type (string) should be the max substring.

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

Input and Output Format:

Input consists of a source string and delimiter.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

delhi-pune-patna

-

Sample Output 1:

Delhi

```
importjava.util.HashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
```

```
Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
System.out.println(User.extractMax(s1,s2));

}
}
```

```
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
import java.util.StringTokenizer;


publicclass User {
publicstatic String extractMax(String s1,String s2){
      StringTokenizer st= new StringTokenizer(s1,s2);
      int max=0,c=0;
      String str=null;
      while(st.hasMoreTokens())
      {
            String s= st.nextToken();
            c=s.length();
            if(c>max)
            {
                  c=max;
            str=s;
            }

      }
      return str;

      }
}
```

15. States and Capitals

Write a program that construts a hashmap with “state” as key and “capital” as its value. If the next input is a state, then it should return capital$state in lowercase.

Include a class UserMainCode with a static method **getCapital** which accepts a hashmap. The return type is the string as given in the above statement

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

Input and Output Format:

Input consists of 2n+2 values. The first value corresponds to size of the hashmap. The next n pair of numbers contains the state and capital. The last value consists of the “state” input.

Output consists of a string as mentioned in the problem statement.

Refer sample output for formatting specifications.

Sample Input 1:

3

Karnataka

Bangaluru

Punjab

Chandigarh

Gujarat

Gandhinagar

Punjab

Sample Output 1:

chandigarh$punjab

```
 import java.util.HashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
HashMap<String,String> hm= new HashMap<String,String>();
for(int i=0;i<n;i++)
      hm.put(sc.next(),sc.next());
String s=sc.next();
System.out.println(User.getCapital(hm,s));

}
}
```

```
importjava.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
importjava.util.StringTokenizer;
```

```
publicclass User {
publicstatic String getCapital(HashMap<String,String> hm,String s){
	Iterator<String> it=hm.keySet().iterator();
	StringBuffer sb= new StringBuffer();
	while(it.hasNext())
	{
		String state=it.next();
		String cap=hm.get(state);
		if(state.equalsIgnoreCase(s))
		{
			sb.append(cap).append('$').append(state);
		}
	}
	return sb.toString().toLowerCase();

	}
}
```

16. Simple String Manipulation - II

Write a program to read a string and return an integer based on the following rules.

If the first word and the last word in the String match, then return the number of characters in the word else return sum of the characters in both words. Assume the Strings to be case - sensitive.

Include a class UserMainCode with a static method **calculateWordSum** which accepts a string. The return type (integer) should be based on the above rules.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

COGNIZANT TECHNOLOGY SOLUTIONS COGNIZANT

Sample Output 1:

9

Sample Input 2:

HOW ARE YOU

Sample Output 2:

6

```
importjava.util.HashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);

String s=sc.nextLine();
System.out.println(User.calculateWordSum (s));

}
}
```

```
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
import java.util.StringTokenizer;


publicclass User {
publicstaticint calculateWordSum (String s){
	int sum=0,i=0;
	StringTokenizer st=new StringTokenizer(s," ");
	String[] s1= new String[st.countTokens()];
	while(st.hasMoreTokens())
	{
	s1[i]=st.nextToken();
	i++;
	}
	if(s1[0].equals(s1[s1.length-1]))
		sum=s1[0].length();
	else
		sum=s1[0].length()+s1[s1.length-1].length();
	return sum;
	}
}
```

17. Vowels, Arrays & ArrayLists

Write a program to read an array of strings and return an arraylist which consists of words whose both first and last characters are vowels. Assume all inputs are in lowecase.

Include a class UserMainCode with a static method **matchCharacter** which accepts a string array. The return type shoud be an arraylist which should contain elements as mentioned above.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' string correspond to the elements in the array.

Output consists of strings which are elements of arraylist

Refer sample output for formatting specifications.

Sample Input 1:

4

abcde

pqrs

abci

orto

Sample Output 1:

abcde

abci

orto

```
import java.util.HashMap;
import java.util.Scanner;
 public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
String[] s= new String[n];
for(int i=0;i<n;i++)
      s[i]=sc.next();
System.out.println(User.matchCharacter  (s));
}
```

```
}

import java.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstatic ArrayList<String> matchCharacter  (String[] s){
      ArrayList<String> a= new ArrayList<String>();
      for(int i=0;i<s.length;i++)
      {
            System.out.println(s[i].charAt(0));
            System.out.println(s[i].charAt(s[i].length()-1));
            if((s[i].charAt(0)=='a'|| s[i].charAt(0)=='e'||
                        s[i].charAt(0)=='i'||s[i].charAt(0)=='o'||
                        s[i].charAt(0)=='u') && (s[i].charAt(s[i].length()-
1)=='a'||
                                          s[i].charAt(s[i].length()-
1)=='e'||s[i].charAt(s[i].length()-1)=='i'||
                                          s[i].charAt(s[i].length()-
1)=='o'||s[i].charAt(s[i].length()-1)=='u'))
            {
                  a.add(s[i]);
            }
      }
      return a;
      }
}
```

18.Transfer from Hashmap to Arraylist

Write a program that constructs a hashmap with “employee id” as key and “name” as its value. Based on the rules below, on being satisfied, the name must be added to the arraylist.

i)First character should be small and the last character should be Capital.

ii)In name at least one digit should be there.

Include a class UserMainCode with a static method **getName** which accepts a hashmap. The return type is an arraylist as expected in the above statement

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

Input and Output Format:

Input consists of 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers contains the employee id and name.

Output consists of arraylist of strings as mentioned in the problem statement.

Refer sample output for formatting specifications.

Sample Input 1:

4

1

ravi5raJ

2

sita8gitA

3

ram8sitA

4

rahul

Sample Output 1:

ravi5raJ

sita8gitA

ram8sitA

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;



public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);
```

```
int n=sc.nextInt();

HashMap<Integer,String> hm= new HashMap<Integer,String>();

ArrayList<String> a= new ArrayList<String>();

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

        hm.put(sc.nextInt(), sc.next());

a=User.getName(hm);

for(int i=0;i<a.size();i++)

{

        System.out.println(a.get(i));

}

}

}
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstatic ArrayList<String> getName(HashMap<Integer,String> hm){
      ArrayList<String> a= new ArrayList<String>();
      Iterator<Integer> it=hm.keySet().iterator();
      while(it.hasNext())
      {
            int id=it.next();
            String name=hm.get(id);
            for(int i=0;i<name.length();i++)
            {
            if(name.charAt(0)>=97 && name.charAt(0)<=122 &&
                        name.charAt(name.length()-1)>=65 &&
name.charAt(name.length()-1)<=96)
            {
                  if(name.charAt(i)>='0'&& name.charAt(i)<='9')
                  {
                        a.add(name);
```

```
                        }
                }

                }
        }
        return a;
        }
}
```

19. Max Admissions

Write a program that reads details about number of admissions per year of a particular college, return the year which had maximum admissions. The details are stored in an arraylist with the first index being year and next being admissions count.

Include a class UserMainCode with a static method **getYear** which accepts a arraylist. The return type is an integer indicating the year of max admissions.

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

Input and Output Format:

Input consists of 2n+1 values. The first value corresponds to size of the data (year & admissions). The next n pair of numbers contains the year and admissions count.

Output consists of an integer as mentioned in the problem statement.

Refer sample output for formatting specifications.

Sample Input 1:

4

2010

200000

2011

300000

2012

45000

2013

25000

Sample Output 1:

2011

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;



public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

ArrayList<Integer> a= new ArrayList<Integer>();

for(int i=0;i<n*2;i++)

        a.add(sc.nextInt());



        System.out.println(User.getYear(a));



}

}
```

```
import java.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;

publicclass User {
publicstaticint getYear(ArrayList<Integer> a){
      intyear=0;
```

```
        int max=0;
        for(int i=1;i<a.size();i=i+2)
        {
                int x=a.get(i);
                if(x>max)
                {
                        max=x;
                        year=a.get(i-1);
                }
        }
        returnyear;
        }
}
```

20. Sum Non Prime Numbers

Write a program to calculate the sum of all the non prime positive numbers less than or equal to the given number.

Note: prime is a natural number greater than 1 that has no positive divisors other than 1 and itself

Example:
input = 9
Prime numbers = 2,3,5 and 7
output = 1+4+6+8+9=28

Include a class **UserMainCode** with a static method "**addNumbers**" that accepts an integer arguement and returns an integer.

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

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

Sample Input:
9

Sample Output:
28

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;



public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

        System.out.println(User.addNumbers(n));



}

}
```

```
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstaticint addNumbers(int n){
      int c=0,sum=0;
      for (int i = 1; i <= n; i++)
     {
        c=0;
for(int j=1;j<=i;j++)
        {
if(i%j==0)
            c++;
        }
      if (c==2)
            ;
      else
        sum=sum+i;
     }
      return sum;
      }
```

```
}
```

---

21. Date Format Conversion

Given a date string in the format dd/mm/yyyy, write a program to convert the given date to the format dd-mm-yy.

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

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

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

Sample Input:
12/11/1998

Sample Output:
12-11-98

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;



public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String n=sc.next();
      System.out.println(User.convertDateFormat(n));

}
```

```
}


import java.text.ParseException;
import java.text.SimpleDateFormat;
importjava.util.ArrayList;
import java.util.Date;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstatic String convertDateFormat(String n) throws ParseException{
      SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
      Date d= sdf.parse(n);
      SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yyyy");
      String s=sdf1.format(d);
      returns;
      }
}
```

22. Valid Date

Given a date string as input, write a program to validate if the given date is in any of the following formats:
dd.mm.yyyy
dd/mm/yy
dd-mm-yyyy

Include a class **UserMainCode** with a static method "**validateDate**" that accepts a String and returns an integer. This method returns 1 if the date is valid, else return -1.

Create a class **Main** which would get a String as input and call the static method **validateDate** 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:
12.03.2012

Sample Output 1:
Valid

Sample Input 2:
27#01#1977

Sample Output 2:
Invalid

```
	publicstaticvoid main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
sdf.setLenient(false);
int res=0;
if(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))
{

	try {
		Date d=sdf.parse(s);
		res=1;
	} catch (ParseException e) {
		res=-1;
	}

System.out.println(res);

	}
}
}
```

23. Convert Format

Given a 10 digit positive number in the format XXX-XXX-XXXX as a string input, write a program to convert this number to the format XX-XX-XXX-XXX.

Include a class **UserMainCode** with a static method "**convertFormat**" that accepts a String argument and returns a String.

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

Input and Output Format:

Input consists of a String.

Output consists of a String.

Sample Input:

555-666-1234

Sample Output:

55-56-661-234

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String n=sc.next();
System.out.println(User.convertFormat(n));

}
}
```

```
import java.text.ParseException;
importjava.text.SimpleDateFormat;
importjava.util.ArrayList;
importjava.util.Date;
importjava.util.HashMap;
importjava.util.Iterator;
import java.util.StringTokenizer;


publicclass User {
publicstatic String convertFormat(String s) throws ParseException{
	StringTokenizer st=new StringTokenizer(s,"-");
	int i=0;
	String[] s1=new String[st.countTokens()];
	while(st.hasMoreTokens())
	{
		s1[i]=st.nextToken();
		i++;
	}
	StringBuffer sb=new StringBuffer();
	sb.append(s1[0].substring(0,2));
		sb.append("-");
		sb.append(s1[0].substring(2)).append(s1[1].substring(0,1));
		sb.append("-");
		sb.append(s1[1].substring(1)).append(s1[2].substring(0,1));
		sb.append("-");
		sb.append(s1[2].substring(1));
```

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

```
import java.util.StringTokenizer;
publicclass UserMainCode {
        publicstatic String convertFormat(String s)
        {
                StringBuffer sb=new StringBuffer();
                StringTokenizer st=new StringTokenizer(s,"-");
                String s1=st.nextToken();
                String s2=st.nextToken();
                String s3=st.nextToken();
           sb.append(s1.substring(0,2));
           sb.append("-");
           sb.append(s1.substring(s1.length()-1));
           sb.append(s2.substring(0,1));
           sb.append("-");
           sb.append(s2.substring(1));
           sb.append(s3.substring(0,1));
           sb.append("-");
           sb.append(s3.substring(1));
        return sb.toString();

        }
}
```

24. Add and Reverse

Given an int array and a number as input, write a program to add all the elements in the array greater than the given number. Finally reverse the digits of the obtained sum and print it.

Include a class **UserMainCode** with a static method "**addAndReverse**" that accepts 2 arguments and returns an integer.The first argument corresponds to the integer array and the second argument corresponds to the number.

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

Example:
Input Array = {10,15,20,25,30,100}
Number = 15
sum = 20 + 25 + 30 + 100 = 175
output = 571

Input and Output Format:
The first line of the input consists of an integer that corresponds to the number of elements in the array.
The next n lines of the input consists of integers that correspond to the elements in the array.
The last line of the input consists of an integer that corresponds to the number.

Output consists of a single integer.

Sample Input
6
10
15
20
25
30
100
15

Sample Output
571

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a=new int[n];
for(int i=0;i<n;i++)
      a[i]=sc.nextInt();
int x=sc.nextInt();
System.out.println(User.addAndReverse(a,x));

}
}
```

```
importjava.text.ParseException;
importjava.text.SimpleDateFormat;
importjava.util.ArrayList;
importjava.util.Date;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstaticint addAndReverse(int[] a,int x) {
      int sum=0;
      int rev=0,r=0;
      for(int i=0;i<a.length;i++)
      {
            if(x<a[i])
                  sum=sum+a[i];
      }

      while(sum!=0)
      {
            r=sum%10;
            rev=rev*10+r;
            sum=sum/10;
      }

            return rev;
      }
}
```

25. Next Year day

Given a date string in dd/mm/yyyy format, write a program to calculate the day which falls on the same date next year. Print the output in small case.

The days are sunday, monday, tuesday, wednesday, thursday, friday and saturday.

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

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

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

Sample Input:
13/07/2012

Sample Output:
saturday

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;



public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String n=sc.next();

System.out.println(User.nextYearDay(n));

}
}
```

```
import java.text.ParseException;
```

```
import java.text.SimpleDateFormat;
importjava.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstatic String nextYearDay(String s) throws ParseException {
      SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
      SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
      Date d= sdf.parse(s);
      Calendar c=Calendar.getInstance();
      c.setTime(d);
    c.add(Calendar.YEAR,1);
    Date year=c.getTime();
    String day=sdf1.format(year);

return day;
}
}
```

26. Sum Squares of Digits

Write a program that accepts a positive number as input and calculates the sum of squares of individual digits of the given number.

Include a class **UserMainCode** with a static method "**getSumOfSquaresOfDigits**" that accepts an integer argument and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **getSumOfSquaresOfDigits** present in the UserMainCode.

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

Sample Input:
321

Sample Output:
14

```
import java.text.ParseException;
```

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

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();

System.out.println(User.getSumOfSquaresOfDigits(n));

}
}
```

```
importjava.text.ParseException;
importjava.text.SimpleDateFormat;
importjava.util.ArrayList;
importjava.util.Calendar;
importjava.util.Date;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstaticint getSumOfSquaresOfDigits(int n) {
	int sum=0,r=0;
	while(n!=0)
	{
		r=n%10;
		sum=sum+(r*r);
		n=n/10;
	}
	return sum;
}
}
```

27. Even and Odd Index Sum

Write a program that accepts a positive number as input and calculates the sum of digits at even indexes (say evenSum) and sum of digits at odd indexes (say oddSum) in the given number. If both the sums are equal , print 'yes', else print no.

Example:
input = 23050
evenSum = 2 + 0 + 0 = 2
oddSum = 3 + 5 = 8
output = no

Include a class **UserMainCode** with a static method "**sumOfOddEvenPositioned**" that accepts an integer and returns an integer. The method returns 1 if the 2 sums are equal. Else the method returns -1.

Create a class **Main** which would get an integer as input and call the static method **sumOfOddEvenPositioned** present in the UserMainCode.

Input and Output Format:
Input consists of an integer.
Output consists of a string that is either "yes" or "no".

Sample Input 1:
23050

Sample Output 1:
no

Sample Input 2:
231

Sample Output 2:
yes

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();

int res=User.sumOfOddEvenPositioned(n);
if(res==1)
      System.out.println("yes");
else
      System.out.println("no");

}
}
importjava.text.ParseException;
importjava.text.SimpleDateFormat;
importjava.util.ArrayList;
```

```
importjava.util.Calendar;
importjava.util.Date;
importjava.util.HashMap;
importjava.util.Iterator;
importjava.util.StringTokenizer;


publicclass User {
publicstaticint sumOfOddEvenPositioned(int n) {
	int even=0,odd=0;
	int res=0,r=0,m=0;
	int j=0,count=0;
int n1=n;
	while(n>0)
	{
		n=n/10;
		count++;
	}
	System.out.println(count);
	int[] a=newint[count];
	while(n1!=0)
	{
		r=n1%10;
		a[j]=r;
		j++;
		n1=n1/10;
	}
	int[] b=newint[j];
	for(int k=j-1;k>=0;k--)
	{
		b[m]=a[k];
		m++;
	}

	for(int i=0;i<m;i++)
	{
		System.out.println("a:"+b[i]);
		if(i%2==0)
			even=even+b[i];
		else
			odd=odd+b[i];
	}
	System.out.println(even);
	System.out.println(odd);
	if(even==odd)
		res=1;
	else
		res=-1;
	return res;
}
}
```

28. Remove 3 Multiples

Write a program that accepts an ArrayList of integers as input and removes every 3rd element and prints the final ArrayList.

Suppose the given arrayList contains 10 elements remove the 3rd, 6th and 9th elements.

Include a class **UserMainCode** with a static method "**removeMultiplesOfThree**" that accepts an ArrayList<Integer> as arguement and returns an ArrayList<Integer>.

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

Input and Output Format:
The first line of the input consists of an integer n, that corresponds to the number of elements to be added in the ArrayList.
The next n lines consist of integers that correspond to the elements in the ArrayList.

Output consists of an ArrayList of integers.

Sample Input:
6
3
1
11
19
17
19

Sample Output
3
1
19
17

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
ArrayList<Integer> a=new ArrayList<Integer>();
ArrayList<Integer> res=new ArrayList<Integer>();
for(int i=0;i<n;i++)
```

```
        a.add(sc.nextInt());
res=User.removeMultiplesOfThree(a);
for(int i=0;i<res.size();i++)
        System.out.println(res.get(i));
}
}

 publicclass User {
publicstatic ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> a)
{
        ArrayList<Integer> b=new ArrayList<Integer>();
        for(int i=0;i<a.size();i++)
        {
                int d=a.get(i);
                if(d%3!=0)
                {
                        b.add(a.get(i));
                }
        }
        return b;
}
}
```

29.String Occurances - II

Obtain two strings S1,S2 from user as input. Your program should count the number of times S2 appears in S1.

Return the count as output. Note - Consider case.

Include a class UserMainCode with a static method **getSubstring** which accepts two string variables. The return type is the count.

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

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of an integer.

Refer sample output for formatting specifications.

Sample Input 1:
catcowcat
cat

Sample Output 1:
2

Sample Input 2:
catcowcat
CAT

Sample Output 2:
0

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);

String s1=sc.next();

String s2=sc.next();

System.out.println(User.getSubstring(s1, s2));

}

}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;



public class User {
public static int getSubstring (String s1,String s2) {
	int count=0;
	int n=s1.length()-(s2.length()-1);
	for(int i=0;i<n;i++)
	{
		String s3=s1.substring(i,i+(s2.length()));
		if(s2.equals(s3))
		count++;
	}
	return count;
}
}
```

```
publicclass User {
publicstaticint getSubstring (String s1,String s2) {
	int count=0;
	int n=s1.length()-(s2.length()-1);
	ints2l=s2.length();
	System.out.println(n);
	for(int i=0;i<n;i++)
	{
		String s3=s1.substring(i,i+s2l);
		if(s2.equals(s3))
		count++;
	}
	return count;
}
}
```

30. Programming Logic

Write a Program that accepts three integer values (a,b,c) and returns their sum. However, if one of the values is 13 then it does not count towards the sum and the next number also does not count. So for example, if b is 13, then both b and c do not count.

Include a class UserMainCode with a static method **getLuckySum** which accepts three integers. The return type is integer representing the sum.

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

Input and Output Format:

Input consists of three integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

Sample Input 1:

1
2
3

Sample Output 1:
6

Sample Input 2:
1
2
13

Sample Output 2:
3

Sample Input 3:
13
3
8

Sample Output 3:
8

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;


public class User {
public static int getLuckySum  (int a,int b, int c) {
```

```
        int res=0;
        if(a==13)
                res=c;
        else if(b==13)
                res=a;
        else if(c==13)
                res=a+b;
        else
                res=a+b+c;

                return res;

}
}
```

---

31. Triplets

Given an integer array, Write a program to find if the array has any triplets. A triplet is a value if it appears 3 consecutive times in the array.

Include a class UserMainCode with a static method **checkTripplets** which accepts an integer array. The return type is boolean stating whether its a triplet or not.

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

Input and Output Format:

Input consists of n+1 integers. The first integer would represent the size of array and the next n integers would have the values.

Output consists of a string stating TRUE or FALSE.

Refer sample output for formatting specifications.

Sample Input 1:

7
3
3
5
5
5
2
3

Sample Output 1:

TRUE

Sample Input 2:

7
5
3
5
1
5
2
3

Sample Output 2:

FALSE

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {
```

```
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a=new int[n];
for(int i=0;i<n;i++)
        a[i]=sc.nextInt();
boolean b=User.checkTripplets(a);
System.out.println(b);
}
}
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;


public class User {
public static boolean checkTripplets  (int a[]) {
boolean b=false;
int c=0;
```

```
for(int i=0;i<a.length-2;i++)
{
        if(a[i]==a[i+1]&&a[i+1]==a[i+2])
                b=true;
                else
                        b=false;
}
return b;
}
}
```

32. Repeat Front

Given a string (s) and non negative integer (n) apply the following rules.

1. Display the first three characters as front.
2. If the length of the string is less than 3, then consider the entire string as front and repeat it n times.

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

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:
Coward
2

Sample Output 1:
CowCow

Sample Input 2:
So
3

Sample Output 2:
SoSoSo

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);


String s=sc.next();

int n=sc.nextInt();

String res=User.repeatFirstThreeCharacters(s,n);
```

```
for(int i=0;i<n;i++)
        System.out.print(res);

}
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
public static String repeatFirstThreeCharacters(String s, int n) {

        String front=null;
        if(s.length()>=3)
        {
                front=s.substring(0,3);
```

```
        }
        else
                front=s;
        return front;
}
}
```

33. Sorted Array

Write a program to read a string array, remove duplicate elements and sort the array.
Note:

1. The check for duplicate elements must be case-sensitive. (AA and aa are NOT duplicates)
2. While sorting, words starting with upper case letters takes precedence.

Include a class UserMainCode with a static method **orderElements** which accepts the string array. The return type is the sorted array.

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

Input and Output Format:

Input consists of an integer n which is the number of elements followed by n string values.

Output consists of the elements of string array.

Refer sample output for formatting specifications.

Sample Input 1:

6
AAA
BBB
AAA
AAA
CCC
CCC

Sample Output 1:

AAA
BBB
CCC

Sample Input 2:

7
AAA
BBB
aaa
AAA
Abc
A
b

Sample Output 2:

A
AAA
Abc
BBB
aaa
b

```
import java.text.ParseException;
```

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

String[] a= new String[n];

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

        a[i]=sc.next();


String res[]=User.orderElements(a);

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

        System.out.println(res[i]);

}

}

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Collections;
```

```
import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.StringTokenizer;



public class User {

public static String[] orderElements(String[] s) {


        LinkedHashSet<String> lhs=new LinkedHashSet<String>();


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

        {

        lhs.add(s[i]);

        }

        String[] a= new String[lhs.size()];

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

        {

                lhs.toArray(a);

        }

        Arrays.sort(a);

        return a;

}
```

}

34. Pattern Matcher

Write a program to read a string and check if it complies to the pattern 'CPT-XXXXXX' where XXXXXX is a 6 digit number. If the pattern is followed, then print TRUE else print FALSE.

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

Output should print TRUE or FALSE .

Refer sample output for formatting specifications.

Sample Input 1:
CPT-302020

Sample Output 1:
TRUE

Sample Input 2:
CPT123412

Sample Output 2:
FALSE

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;
```

```
import java.util.Iterator;

import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);

String a= sc.next();


boolean b=User.CheckID(a);


        System.out.println(b);

}

}

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Calendar;

import java.util.Collections;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.StringTokenizer;
```

```
public class User {

public static boolean CheckID (String s) {

boolean b=false;

if(s.matches("(CPT-)[0-9]{6}"))

                b=true;

else

        b=false;

return b;

}

}
```

35. Playing with String - I

Given a string array and non negative integer (n) apply the following rules.

1. Pick nth character from each String element in the String array and form a new String.
2. If nth character not available in a particular String in the array consider $ as the character.
3. Return the newly formed string.

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

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 an integer which denotes the size of the array followed by the array of strings and

an integer (n).
Output consists of a string .
Refer sample output for formatting specifications.

Sample Input 1:
4
ABC
XYZ
EFG
MN
3

Sample Output 1:
CZG$

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);

int n=sc.nextInt();

String[] a=new String[n];

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

        a[i]=sc.next();

int s=sc.nextInt();
```

```
System.out.println(User.formString(a,s));
}
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;


public class User {
public static String formString(String s[],int n) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s.length;i++)
        {
                String st=s[i];
                if(st.length()>=n)
```

```
                {
                        sb.append(st.charAt(n-1));
                }
                else
                        sb.append("$");
        }
return sb.toString();
}
}
```

36. Regular Expression - 1

Given a string (s) apply the following rules.

1. String should be only four characters long.
2. First character can be an alphabet or digit.
3. Second character must be uppercase 'R'.
4. Third character must be a number between 0-9.

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

Include a class UserMainCode with a static method **validate** 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:
vR4u

Sample Output 1:
TRUE

Sample Input 2:
vRau

Sample Output 2:
FALSE

Sample Input 3:
vrau

Sample Output 3:
FALSE

```
import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);


String s=sc.next();

System.out.println(User.validate (s));
```

```
}
}

public class User {
public static boolean validate (String s) {
        boolean b= false;
        if(s.length()==4)
        {
        if(s.matches("[a-z0-9]{1}(R)[0-9]{1}[A-Za-z0-9]{1}"))
                b=true;
        else
                b=false;
}
        return b;
}
}
```

37. Regular Expression – 2 (Age Validator)

Given the age of a person as string, validate the age based on the following rules.

1. Value should contain only numbers.
2. Value should be non-negative.
3. Value should be in the range of 21 to 45'.

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

Include a class UserMainCode with a static method **ValidateAge** 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:
23

Sample Output 1:
TRUE

Sample Input 2:
-34

Sample Output 2:
FALSE

Sample Input 3:
3a

Sample Output 3:
FALSE

```
import java.util.Scanner;


public class Main {

public static void main(String[] args) throws ParseException {

Scanner sc = new Scanner(System.in);
```

```
int s=sc.nextInt();
System.out.println(User.validate (s));

}
}

public class User {
public static boolean validate (int s) {
        boolean b= false;
        if(s>0)
        {
        if(s>=21&&s<=45)
                b=true;
        else
                b=false;
}
        return b;
}
}
```

38. Regular Expression – 3 (Phone Validator)

Given a phone number as string, validate the same based on the following rules.

1. Value should contain only numbers.

2. Value should contain 10 digits.
3. Value should not start with 00.

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

Include a class UserMainCode with a static method **validatePhone** 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:
9987684321

Sample Output 1:
TRUE

Sample Input 2:
0014623452

Sample Output 2:
FALSE

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;
```

```
public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);

String s=sc.next();
System.out.println(User.validatePhone(s));

}
}

public class User {
public static boolean validatePhone(String s) {
	boolean b= false;
	if(s.length()==10)
	{
	if(s.matches("(0){2}[0-9]{8}"))
		b=false;
	else if(s.matches("[0-9]{10}"))
		b=true;
	else
		;
}
	return b;
}
```

}

39. String Splitter

Write a program which would accept a string and a character as a delimiter. Apply the below rules

1. Using the delimiter, split the string and store these elements in array.
2. Reverse each element of the string and convert it into lowercase.

Include a class UserMainCode with a static method **manipulateLiteral** which accepts the string and character. The return type is the string array formed.

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

Input and Output Format:

Input consists of a string and character.
Output consists of a string array.

Refer sample output for formatting specifications.

Sample Input 1:
AAA/bba/ccc/DDD
/

Sample Output 1:
aaa
abb
ccc
ddd

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);

String s1=sc.next();
char s2=sc.next().charAt(0);
```

```
String res[]=User.manipulateLiteral (s1,s2);
for(int i=0;i<res.length;i++)
      System.out.println(res[i]);

}
}


publicclass User {
publicstatic String[] manipulateLiteral(String s1,char s2) {
      String ss=Character.toString(s2);
      StringTokenizer st=new StringTokenizer(s1,ss);
      ArrayList<String> a=new ArrayList<String>();
      while(st.hasMoreTokens())
      {
            StringBuffer sb=new StringBuffer();
            sb.append(st.nextToken().toLowerCase());
            a.add(sb.reverse().toString());

      }
      String[] s=new String[a.size()];
      for(int i=0;i<a.size();i++)
       s[i]=(String)a.get(i);


return s;
}
}
```

```
import java.util.ArrayList;
import java.util.StringTokenizer;

publicclass User {
publicstatic String[] manipulateLiteral(String s1,char s2) {
      String ss=String.valueOf(s2);
      StringTokenizer st=new StringTokenizer(s1,ss);
      ArrayList<String>a=new ArrayList<String>();

      while(st.hasMoreTokens())
      {           StringBuffer sb=new StringBuffer();
            sb.append(st.nextToken());
            a.add(sb.reverse().toString().toLowerCase());

      }
      int d=a.size();
      System.out.println(d);
      String[] s=new String[d];
      for(int i=0;i<a.size();i++)
      {
       s[i]=a.get(i);
      }
```

```
return s;
}
}
```

40. Vowel Count

Write a program to read a string and count the number of vowels present in it.

Include a class UserMainCode with a static method **tellVowelCount** which accepts the string. The return type is the integer giving out the count of vowels.

Note: The check is case-insensitive.

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 integer.
Refer sample output for formatting specifications.

Sample Input 1:
NewYork

Sample Output 1:
2

Sample Input 2:
Elephant

Sample Output 2:
3

```
import java.util.Scanner;

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);

String s1=sc.next();

System.out.println(User.tellVowelCount(s1));
}
}

publicclass User {
publicstaticint tellVowelCount(String s1) {
	int count=0;
	String s="aeoiu";
	String ss="AEIOU";
	for(int i=0;i<s1.length();i++)
	{
		for(int j=0;j<s.length();j++)
		{
			if(s1.charAt(i)==s.charAt(j) || s1.charAt(i)==ss.charAt(j)
)
				count++;
		}
	}
	return count;
}
}
```

```
publicstaticint tellVowelCount(String s1) {
	int count=0;
	for(int i=0;i<s1.length();i++)
	{
			if(s1.charAt(i)=='a' || s1.charAt(i)=='e' ||
s1.charAt(i)=='i'
				|| s1.charAt(i)=='o' || s1.charAt(i)=='u' ||
				s1.charAt(i)=='A' || s1.charAt(i)=='E' ||
				s1.charAt(i)=='I' ||
				s1.charAt(i)=='O' || s1.charAt(i)=='U' )

			{
				count++;
			}
		}

	return count;
}
}
```

****

41. Playing with String - II

Write a program to accept a string array as input, convert all the elements into lowercase and sort the string array. Display the sorted array.

Include a class UserMainCode with a static method **sortArray** which accepts the string array. The return type is the string array formed based on requirement.

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

Input and Output Format:

Input consists of a an integer which denotes the size of the array followed by the array of strings,
Output consists of a string array.
Refer sample output for formatting specifications.

Sample Input 1:
5
AAA
BB
CCCC
A
ABCDE

Sample Output 1:
a
aaa
abcde
bb
cccc

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
String[] a= new String[n];
for(int i=0;i<n;i++)
      a[i]=sc.next();
String[] res=User.sortArray(a);
for(int i=0;i<res.length;i++)
      System.out.println(res[i]);
```

```
}
}
```

```
publicclass User {
publicstatic String[] sortArray (String s[]) {
      String[] a=new String[s.length];
for(int i=0;i<s.length;i++)
{
a[i]=s[i].toLowerCase();
}
Arrays.sort(a);
return a;
}
}
```

42. Median Calculation

Write a program to accept an int array as input, and calculate the median of the same.

Median Calculation Procedure:

1. Sort the sequence of numbers.
2. The total number count is odd, Median will be the middle number.

The total number count is even, Median will be the average of two middle numbers, After calculating the average, round the number to nearest integer.

Include a class UserMainCode with a static method **calculateMedian** which accepts the int array. The return type is the integer which would be the median.

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

Input and Output Format:

Input consists of a an integer which denotes the size of the array followed by the array of integers.
Output consists of a integer.
Refer sample output for formatting specifications.

Sample Input 1:

7

1

2
1
4
7
1
2

Sample Output 1:

2

Sample Input 2:

6
52
51
81
84
60
88

Sample Output 2:

71

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
      a[i]=sc.nextInt();
System.out.println(User.calculateMedian (a));

}
}



publicclass User {

publicstaticint calculateMedian(int s[]) {
      double med=0;
      double avg=0;
Arrays.sort(s);
int mid=s.length/2;
if(s.length%2!=0)
      med=s[mid];
else
{
```

```
avg=(double)(s[mid]+s[mid-1]\)/2;
     System.out.println(avg);
       med=Math.ceil(avg);
}
return(int)med;
}
}
```

43. Sequence in Array

Write a program to accept an int array as input, and check if [1,2,3] appears somewhere in the same sequence.

Include a class UserMainCode with a static method **searchSequence** which accepts the int array. The return type is a boolean which returns true or false.

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

Input and Output Format:

Input consists of a an integer which denotes the size of the array followed by the array of integers.
Output should print true or false.
Refer sample output for formatting specifications.

Sample Input 1:
9
11
-2
5
1
2
3
4
5
6

Sample Output 1:
TRUE

Sample Input 2:
6
-2
5
1
3
2
6

Sample Output 2:

FALSE

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
      a[i]=sc.nextInt();
boolean b=User.calculateMedian (a);
System.out.println(b);

}
}

publicclass User {
publicstaticboolean calculateMedian(int s[]) {
      int[] a={1,2,3};
      int n=s.length-(a.length-1);
      boolean b=false;
      for(int i=0;i<n;i++)
      {
            if(s[i]==a[0] )
            {
                  if(s[i+1]==a[1])
                  {
                        if(s[i+2]==a[2])
                        {
                              b=true;
                              break;
                        }
                        else
                              b=false;
            }
                  else
                        b=false;
            }
            else
                  b=false;
      }
      return b;
}
}
```

44. Asterisk & Characters

Write a program to read a string and return true or false based on the below rule:

1. Return true if for every '*' in the string, there are same characters both side immediately before and after the star, else return false.

Include a class UserMainCode with a static method **scanStarNeighbors** which accepts the string. The return type is the boolean TRUE or FALSE based on the rule.

Note: The check is case-insensitive.

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:
Hello*World

Sample Output 1:
FALSE

Sample Input 2:
Welcome*elizabeth

Sample Output 2:
TRUE

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String n=sc.next();

boolean b=User.scanStarNeighbors(n);
System.out.println(b);

}
}
```

```
publicclass User {
publicstaticboolean scanStarNeighbors(String s) {
	StringTokenizer st=new StringTokenizer(s,"*");
	boolean b=false;
	while(st.hasMoreTokens())
	{
		String s1=st.nextToken();
		String s2=st.nextToken();
		if(s1.charAt(s1.length()-1)==s2.charAt(0))
		{
			b=true;
		}
	}
	return b;
}
}
```

45. Occurance Count

Write a program to read a string that contains a sentence and read a word. Check the number of occurances of that word in the sentence.

Include a class UserMainCode with a static method **countWords** which accepts the two strings. The return type is the integer giving the count.

Note: The check is case-sensitive.

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

Input and Output Format:

Input consists of two strings.
Output consists of count indicating the number of occurances.
Refer sample output for formatting specifications.

Sample Input 1:
Hello world Java is best programming language in the world
world

Sample Output 1:
2

Sample Input 2:

hello world

World

Sample Output 2:

0

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.next();
intb=User.countWords (s1,s2);
System.out.println(b);

}
}
```

```
publicclass User {
publicstaticint countWords (String s1,String s2) {
	StringTokenizer st=new StringTokenizer(s1," ");
	int c=0;
	while(st.hasMoreTokens())
	{
		String s3=st.nextToken();
	if(s3.equals(s2))
		{
			c++;
		}
	}
	return c;
}
}
```

46.Regular Expressions - III

Write a program to read two strings S1 & S2, compute the number of times that S2 appears in S1.

Include a class UserMainCode with a static method **searchString** which accepts the two strings. The return type is the integer giving the count.

Note: The check is case-insensitive.

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

Input and Output Format:

Input consists of two strings.

Output consists of count indicating the number of occurances.

Refer sample output for formatting specifications.

Sample Input 1:

Catcowcat

cat

Sample Output 1:

2

Sample Input 2:

Catcowcat

catp

Sample Output 2:

0

```
publicclass User {
	publicstaticint scanStarNeighbors(String s1,String s2) {
		int ls1=s1.length();
		int ls2=s2.length();
		int n=ls1-(ls2-1);
		System.out.println(n);
		int ct=0;
		for(int i=0;i<n;i++)
		{

			String ss=s1.substring(i,i+(ls2));
			if(s2.equals(ss))
			ct++;
			}

		return ct;
	}
	}
```

47. Strings Processing

Write a program to read a string that contains comma separated fruit names and also a number N. Pick the nth fruit and return it. If the total number of elements are less than the number specified in N, then return the last element.

Include a class UserMainCode with a static method **findFruitName** which accepts the the string and the number n. The return type is the string which has the fruit name.

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:
Apple,Banana,Orange
2

Sample Output 1:
Banana

Sample Input 2:
Apple,Banana,Orange
4

Sample Output 2:
Orange

```
import java.text.ParseException;
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.Iterator;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
```

```
String s1=sc.nextLine();
int n=sc.nextInt();
System.out.println(User.findFruitName(s1,n));
}
}

importjava.util.ArrayList;
importjava.util.Arrays;
importjava.util.LinkedHashSet;
import java.util.StringTokenizer;
publicclass User {
publicstatic String findFruitName(String s1,int n) {
	StringTokenizer st=new StringTokenizer(s1,",");
	intc=0,i=0;
	String ss=null;

	String[] s=new String[st.countTokens()];

	while(st.hasMoreTokens())
	{
		s[i]=st.nextToken();
		i++;
	}
	if(i>n)
	{
	 ss=s[n-1];
	}
	else
	{
		ss=s[i-1];//last element display
	}

	return ss;
}
}
```

48. Proper Case

Write a program to read a string and convert the intial letter of each word to uppercase.

Include a class UserMainCode with a static method **changeCase** 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:
This is cognizant academy

Sample Output 1:
This Is Cognizant Academy

49. Length of same word

Write a program to read a string containing multiple words find the first and last words, if they are same, return the length and if not return the sum of length of the two words.

Include a class UserMainCode with a static method **compareLastWords** which accepts the string. The return type is the length as per problem.

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 integer.
Refer sample output for formatting specifications.

Sample Input 1:
This is Cognizant Academy

Sample Output 1:
11

Sample Input 2:

Hello World Hello

Sample Output 2:

5

```
import java.util.StringTokenizer;
publicclass UserMainCode {
	publicstatic String changeWord(String s)
	{
		StringTokenizer st=new StringTokenizer(s," ");
		StringBuffer sb=new StringBuffer();
		while(st.hasMoreTokens())
		{
			String s1=st.nextToken();
			sb.append(s1.substring(0,1).toUpperCase());
			sb.append(s1.substring(1));
			sb.append(" ");

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

50. Perfect Number

Write a program to that takes a positive integer andreturns true if the number is perfect number.

A positive integer is called a perfect number if the sum of all its factors (excluding the number itself, i.e., proper divisor) is equal to its value.

For example, the number 6 is perfect because its proper divisors are 1, 2, and 3, and 6=1+2+3; but the number 10 is not perfect because its proper divisors are 1, 2, and 5, and 1+2+5 is not equal to 10

Include a class UserMainCode with a static method **getPerfection** which accepts the number. The

return type is boolean (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 integer.
Output consists of TRUE / FALSE.
Refer sample output for formatting specifications.

Sample Input 1:
28

Sample Output 1:
TRUE

```
publicclass User {
	publicstaticboolean scanStarNeighbors(intn) {
		boolean b;

		int sum=0;

			for(int i=1;i<n;i++)
			{
				if(n%i==0)
				{
				 sum=sum+i;
				 System.out.println(sum);
				}

			}
			if(sum==n)
			{
				b=true;
			}
			else
			{
				b=false;
			}


		return b;
	}
```

```
                    }
```

---

51. Find Digits

For a given double number with atleast one decimal value, Write a program to compute the number of digits before and after the decimal point in the following format – noOfDigitsBeforeDecimal:noOfDigitsAfterDecimal.

Note: Ignore zeroes at the end of the decimal (Except if zero is the only digit after decimal. Refer Example 2 and 3)

Include a class UserMainCode with a static method **findNoDigits** which accepts the decimal value. The return type is 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 double.
Output consists of string.
Refer sample output for formatting specifications.

Sample Input 1:
843.21

Sample Output 1:
3:2

Sample Input 2:
20.130

Sample Output 2:
2:2

Sample Input 3:
20.130

Sample Output 3:

2:2

```
import java.util.*;

public class Main {

public static void main(String[] args) {

double d=845.69;

System.out.println(noOfDigits(d));

}

public static String noOfDigits(double d) {

int n1=0,n2=0;

String s=String.valueOf(d);

StringTokenizer t=new StringTokenizer(s,".");

String s1=t.nextToken();

String s2=t.nextToken();

n1=s1.length();

n2=s2.length();

if(s1.charAt(0)=='0')

n1=s1.length()-1;

if(n2!=1)

if(s2.charAt(s2.length()-1)=='0')

n2=s2.length()-1;

String s3=String.valueOf(n1)+":"+String.valueOf(n2);

return s3;

}
```

}

```
import java.util.StringTokenizer;
publicclass User{
publicstatic String noOfDigits(double d) {
int n1=0,n2=0;
String s=String.valueOf(d);
StringTokenizer t=new StringTokenizer(s,".");
String s1=t.nextToken();
String s2=t.nextToken();
n1=s1.length();
n2=s2.length();
if(s1.charAt(0)=='0')
n1=s1.length()-1;
//if(n2!=1)
if(s2.charAt(n2-1)=='0')
n2=s2.length()-1;
//String s3=String.valueOf(n1)+":"+String.valueOf(n2);
StringBuffer sb=new StringBuffer();
sb.append(n1).append(":").append(n2);
return sb.toString();
}
}
```

52. Employees & Designations

A Company wants to obtain employees of a particular designation. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

Read Employee details from the User. The details would include name and designaton in the given order. The datatype for name and designation is string.

Build a hashmap which contains the name as key and designation as value.

You decide to write a function **obtainDesignation** which takes the hashmap and designation as input and returns a string List of employee names who belong to that designation as output. Include this function in class UserMainCode. Display employee name's in ascending order.

Create a Class Main which would be used to read employee details in step 1 and build the hashmap. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of employee details. The first number indicates the size of the employees. The next two values indicate the employee name employee designation. The last string would be the designation to be searched.

Output consists of a array values containing employee names.
Refer sample output for formatting specifications.

Sample Input 1:
4
Manish
MGR
Babu
CLK
Rohit
MGR
Viru
PGR
MGR

Sample Output 1:
Manish
Rohit

```
class Main
{
	public static void main(String[] arg)
	{

	Scanner sc=new Scanner(System.in);
	int n=sc.nextInt();
		sc.nextLine();
		HashMap<String,String> hm=new HashMap<String,String>();
```

```
        for(int i=0;i<n;i++)
        {
                hm.put(sc.nextLine(),sc.nextLine());
        }
                String b=sc.nextLine();

                HashMap<String,String> op=new HashMap<String,String>();
                op=MainClass.obtainDesig(hm,b);
                Iterator<String> itr=op.keySet().iterator();
                while(itr.hasNext())
                {
                        String key=itr.next();
                                System.out.println(key);
                                String value=hm.get(key);
                                System.out.println(value);

                }
}}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;

public class MainClass {
	public static  HashMap<String,String> obtainDesig(HashMap<String,String> hm,String s)
	{
		LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();
		Iterator<String> itr=hm.keySet().iterator();

		while(itr.hasNext())
		{
			String key=itr.next();
			String value=hm.get(key);
			if(s.equals(value))
			{
				op.put(key,value);
			}


		}
		return op;
			}
}
```

53.Grade Calculator

A School wants to give assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

Read student details from the User. The details would include name, mark in the given order. The datatype for name is string, mark is float.

You decide to build a hashmap. The hashmap contains name as key and mark as value.

BUSINESS RULE:

1. If Mark is less than 60, then grade is FAIL.
2. If Mark is greater than or equal to 60, then grade is PASS.

Note: FAIL/PASS should be in uppercase.

Store the result in a new Hashmap with name as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmap as input and returns the hashmap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read student details in step 1 and build the hashmap. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of student details. The first number indicates the size of the students. The next two values indicate the name, mark.

Output consists of a name and corresponding grade for each student.

Refer sample output for formatting specifications.

Sample Input 1:

3
Avi
76.36
Sunil
68.42
Raja
36.25

Sample Output 1:

Avi
PASS
Sunil
PASS
Raja
FAIL

```
import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;


public class Main


{

public static void main(String[]arg)

{

LinkedHashMap<String,Double>hm=new LinkedHashMap<String,Double>();

LinkedHashMap<String,String>hm1=new LinkedHashMap<String,String>();

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

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

{

String s=sc.next();

double d=sc.nextDouble();
```

```
hm.put(s,d);
}
LinkedHashMap<String,String>hm2=UserMainCode.dis(hm);
for(Map.Entry<String,String>entry:hm2.entrySet())
{
System.out.println(entry.getKey());
System.out.println(entry.getValue());
}}}
```

```
import java.util.LinkedHashMap;
import java.util.Map;

class UserMainCode
{
public static LinkedHashMap<String,String>dis(LinkedHashMap<String,Double>h1)
{
        LinkedHashMap<String,String>h2=new LinkedHashMap<String,String>();
for(Map.Entry m:h1.entrySet())
{
double d=(Double)m.getValue();
if(d>60)
{
```

String s=(String)m.getKey();

h2.put(s,"pass");

}

else

{

String s=(String)m.getKey();

h2.put(s,"fail");

}

}

return h2;

}

}

(Or)

```
import java.util.*;
publicclass Main {
publicstaticvoid main(String[] args) {
	Scanner sc=new Scanner(System.in);

int n=sc.nextInt();
LinkedHashMap<String,Float> ip=new LinkedHashMap<String,Float>();
for(int i=0;i<n;i++)
{
	ip.put(sc.next(),sc.nextFloat());
}

LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();
op=User.noOfDigits(ip);
Iterator<String> itr= op.keySet().iterator();
while(itr.hasNext())
{
	String key=itr.next();
	System.out.println(key);
	String value=op.get(key);
	System.out.println(value);

}
}}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
publicclass User{
publicstatic LinkedHashMap<String,String>  noOfDigits(HashMap<String,Float>
hm) {

	LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();
	Iterator<String> itr=hm.keySet().iterator();
	String res=null;
		for(int i=0;i<hm.size();i++)
		{
	while(itr.hasNext())
	{
		String key=itr.next();

		float value=hm.get(key);
		if(value>=60)
			res="pass";
		else
			res="fail";
		op.put(key,res);


	}
		}

	return op;
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
publicclass User{
publicstatic LinkedHashMap<String,String>  noOfDigits(HashMap<String,Float>
hm) {

	LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();
	Iterator<String> itr=hm.keySet().iterator();
	String res=null;
			while(itr.hasNext())
	{
		String key=itr.next();

		float value=hm.get(key);
		if(value>=60)
			res="pass";
		else
			res="fail";
		op.put(key,res);

	}
```

```
        return op;
}
}
```

54. DOB - Validation

Write a program to validate the Date of Birth given as input in String format (MM/dd/yyyy) as per the validation rules given below. Return true for valid dates else return false.
1. Value should not be null
2. month should be between 1-12, date should be between 1-31 and year should be a four digit number.
Include a class UserMainCode with a static method **ValidateDOB** 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:
12/23/1985

Sample Output 1:
TRUE

Sample Input 2:
31/12/1985

Sample Output 2:
FALSE

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class UserMainCode {
public static void main(String[] args)
{
String str=new String();
Scanner sc=new Scanner(System.in);
str=sc.nextLine();
SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");
sdf.setLenient(false);
try
{
Date d1=sdf.parse(str);
System.out.println("TRUE");
}
catch(Exception e)
{
System.out.println("FALSE");
}
}
}
```

55. Experience Validator

Write a program to valiadate the experience of an employee.

An employee who has recently joined the organization provides his year of passing and total number of years of experience in String format. Write code to validate his experience against the current date.

1) Input consists of two String first represent the year of passed out and the second string reperesent the year of experience.
2) create a function with name **validateExp** which accepts two string as input and boolean as output.
3) The difference between current year and year of pass should be more than or equal to Experience
Return true if all condition are true.

Note:Conside 2015 as the current year.

Include a class UserMainCode with the static function validateExp

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

Input and Output Formate:
Input consists of two Strings.
output will display true if the given data are correct.

Sample Input:
2001
5

Sample Output:
TRUE

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;


public class Main {

    public static void main(String args[]){

       Scanner sc = new Scanner(System.in);


      String s=sc.nextLine();

      String s1=sc.nextLine();


     System.out.print(UserMainCode.getvalues(s,s1));

    }

}

import java.util.Calendar;


import java.util.Date;




public class UserMainCode {

            public static boolean getvalues(String s,String s1)

            {

                        int y1=Integer.parseInt(s);

                        Date d=new Date();
```

```
Calendar c=Calendar.getInstance();
int y2=c.get(Calendar.YEAR);


int y=Math.abs(y1-y2);
int e=Integer.parseInt(s1);
if(y>=e)
        return true;
else
        return false;


}}
```

56. ArrayList to String Array

Write a program that performs the following actions:

Read n strings as input.

Create an arraylist to store the above n strings in this arraylist.

Write a function convertToStringArray which accepts the arraylist as input.

The function should sort the elements (strings) present in the arraylist and convert them into a string array.

Return the array.

Include a class UserMainCode with the static method **convertToStringArray** which accepts an arraylist and returns an array.

Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.

Input and Output Format:

Input consists of n+1 integers. The first integer denotes the size of the arraylist, the next n strings are values to the arraylist.

Output consists of an arrayas per step 4.
Refer sample output for formatting specifications.

Sample Input 1:

4
a
d
c
b

Sample Output 1:

a
b
c
d

```
import java.util.ArrayList;

import java.util.Scanner;


class Main

{

	public static void main(String[] arg)

	{

		Scanner sc=new Scanner(System.in);

		int n=sc.nextInt();

		sc.nextLine();


		ArrayList<String> aa=new ArrayList<String>();
```

```
		for(int i=0;i<n;i++)
		{
			aa.add(sc.nextLine());
		}

		String a[]=MainClass.convertToString( aa);
		for(int i=0;i<a.length;i++)
		{
			System.out.println(a[i]);
		}


	}}

import java.util.ArrayList;
import java.util.Collections;


public class MainClass {


	public static  String[] convertToString(ArrayList<String> a1)
	{
		Collections.sort(a1);// uses to sort arraylist string
```

```
        String a[]=new String[a1.size()];
        a1.toArray(a);
        return a;

}
}
```

57. State ID generator

Write a program to generate the state ID.

1)Read n Strings as input(as State Name).
2)Create a String Array to Store the above Input.
3)Write a function **getStateId** which accepts String Array as input.
4)Create a HashMap<String,String> which stores state name as key and state Id as Value.
5)The function getStateId returns the HashMap to the Main Class.

Include UserMainCode Class With static method **getStateId** which accepts String array and return a hashmap.

Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.

Input and Output Format:
Input Consists of an integer n denotes the size of the string array.
Output consists of an HashMap displayed in the string array order.

Sample Input 1:
3
Kerala
Gujarat
Goa

Sample Output 1:

KER:Kerala

GUJ:Gujarat

GOA:Goa

```
import java.util.*;
publicclass Main
{
	publicstaticvoid main(String[] args)
	{

	Scanner sc=new Scanner(System.in);

		int n=sc.nextInt();
		String s1[]=new String[n];
		for(int i=0;i<n;i++)
		{
	 s1[i]=sc.next();
		}


		LinkedHashMap<String,String>ip=new
LinkedHashMap<String,String>();
		ip=User.Method(s1);
		Iterator<String> itr=ip.keySet().iterator();

		//while(itr.hasNext())
		for(int i=0;i<ip.size();i++)
		{
			String key=itr.next();
			String value=ip.get(key);
			System.out.println(value+""+key);
		}

	}
}
```

```
import java.util.LinkedHashMap;
publicclass User
{
publicstatic LinkedHashMap<String,String>  Method(String[] s1)
{
	LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();

	for(int i=0;i<s1.length;i++)
	{
		 StringBuffer sb=new StringBuffer();
	StringBuffer key=sb.append(s1[i].substring(0,3)).append(":");
	op.put(s1[i],key.toString().toUpperCase());

	}
```

```
      return op;
}
}
```

(or)

STATE id

```
  import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;



public class Main {

       public static void main(String[] args) {
              BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
              try
              {
                     int n=Integer.parseInt(br.readLine());
                     String[] input=new String[n];
                     for(int i=0;i<n;i++)
                     {
                           input[i]=br.readLine();
                     }
                     HashMap<String,String>hm=UserMainCode.costEst
                imator(input);
                     for(int i=0;i<n;i++)
                     {
                           String s=input[i];
                           String key=hm.get(s);
                           System.out.println(key+":"+s);
                     }
              }


catch(Exception e)
              {

              }

       }
```

```
}
  import java.util.HashMap;

public class UserMainCode {
       public static HashMap<String,String>
costEstimator(String[] name)
       {
              int n=name.length;
              HashMap<String, String> hm=new HashMap<String,
String>();
              for(int i=0;i<n;i++)
              {
                     String sub=name[i].substring(0, 3);
                     hm.put(name[i],sub.toUpperCase());
              }
              return hm;
       }

}
```

58.ArrayList to String Array

Write a program that performs the following actions:

1.Read m strings as input (fruit names).
2.Create an arraylist to store the above m strings in this arraylist.
3.Read n strings as input (fruit names).
4.Create an arraylist to store the above n strings in this arraylist.
5.Write a function fruitSelector which accepts the arraylists as input.
6.Remove all fruits whose name ends with 'a' or 'e' from first arrayList and remove all fruits whose name begins with 'm' or 'a' from second arrayList then combine the two lists and return the final output as a String array.
7.If the array is empty the program will print as “No fruit found”
Include a class UserMainCode with the static method **fruitSelector** which accepts the two arraylists and returns an array.

Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.

Input and Output Format:

Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.

Output consists of an array as per step 6. Refer sample output for formatting specifications.

Sample Input 1:
3
Apple
Cherry
Grapes
4
Orange
Mango
Melon
Apple
Sample Output 1:
Cherry
Grapes
Orange

```
import java.util.*;
public class Main {
public static void main(String[] args) {
        List<String> l1=new ArrayList<String>();
        l1.add("Apple");
        l1.add("Chery");
        l1.add("Grapes");
        List<String> l2=new ArrayList<String>();
        l2.add("Orange");
        l2.add("Mango");
        l2.add("Melon");
        l2.add("Apple");
        String[] s2=fruitsList(l1,l2);
```

```
        for(String s3:s2)
                System.out.println(s3);
}
public static String[] fruitsList(List<String> l1, List<String> l2){
        List<String> l3=new ArrayList<String>();
        for(int i=0;i<l1.size();i++){
                String s1=l1.get(i);

                if(s1.charAt(s1.length()-1)!='a' && s1.charAt(s1.length()-1)!='A' &&
s1.charAt(s1.length()-1)!='e' && s1.charAt(s1.length()-1)!='E')
                        l3.add(s1); }
        for(int i=0;i<l2.size();i++){
                String s1=l2.get(i);
                if(s1.charAt(0)!='m' && s1.charAt(0)!='M' && s1.charAt(0)!='a' &&
s1.charAt(0)!='A')
                        l3.add(s1); }
        Collections.sort(l3);
        String[] s2=new String[l3.size()];
        for(int i=0;i<s2.length;i++)
                s2[i]=l3.get(i);
        return s2;
}
```

```
import java.util.*;
publicclass Main {
publicstaticvoid main(String[] args) {
      List<String> l1=new ArrayList<String>();
      Scanner sc=new Scanner(System.in);
      int n=sc.nextInt();
      for(int i=0;i<n;i++)
      {
            l1.add(sc.next());
      }
      List<String> l2=new ArrayList<String>();

      int n1=sc.nextInt();
      for(int i=0;i<n1;i++)
      {
            l2.add(sc.next());
      }
```

```
        String[] s2=User.fruitsList(l1,l2);
        // for(int i=0;i<s2.length;i++)
              // System.out.println(s2[i].toString());
        for(String s3:s2)
              System.out.println(s3);

}
}


publicclass User
{
publicstatic String[] fruitsList(List<String> l1, List<String> l2){
        ArrayList<String> l3=new ArrayList<String>();
        for(int i=0;i<l1.size();i++)
        {
              String s1=l1.get(i);
              intlen=s1.length();
        if(s1.charAt(len-1)!='a'&& s1.charAt(len-1)!='A'
              && s1.charAt(len-1)!='e'&& s1.charAt(len-1)!='E')

              l3.add(s1);
        }


        for(int i=0;i<l2.size();i++)
        {
              String s1=l2.get(i);
              if(s1.charAt(0)!='m'&& s1.charAt(0)!='M'&& s1.charAt(0)!='a'
                    && s1.charAt(0)!='A')
                    l3.add(s1);
              }
        Collections.sort(l3);
        String[] s2=new String[l3.size()];
        for(int i=0;i<s2.length;i++)
              s2[i]=l3.get(i);
        return s2;
}

}



]
   /'
```

59. Elements in ArrayList

Use Collection Methods.
Write a program that takes two ArrayLists as input and finds out all elements present either in A or B, but not in both.

Include a class UserMainCode with the static method arrayListSubtractor which accepts the two arraylists and returns an array.

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

Input and Output Format:

Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.

Output consists of an array. The elements in the output array need to be printed in sorted order.

Refer sample output for formatting specifications.

Sample Input 1:
4
1
8
3
5
2
3
5
Sample Output 1:
1
8

Sample Input 2:
4
9

1
3
5
4
1
3
5
6

Sample Output 2:

6
9

```
import java.util.*;
publicclass Main
{
	publicstaticvoid main(String[] args)
	{
		int n,m;
		Scanner sc = new Scanner(System.in);
		n = sc.nextInt();
		ArrayList<Integer> a1 = new ArrayList<Integer>();
		for(int i=0;i<n;i++)
		{

			a1.add(sc.nextInt());
		}
		m = sc.nextInt();
		ArrayList<Integer> a2 = new ArrayList<Integer>();
		for(int i=0;i<m;i++)
		{

			a2.add(sc.nextInt());
		}
		int[] result = User.arrayListSubtractor(a1, a2);
		Arrays.sort(result);
		for(int i=0;i<result.length;i++)
			System.out.println(result[i]);
}
}
```

```
import java.util.*;

publicclass User
{


	publicstaticint[] arrayListSubtractor(ArrayList<Integer>
	arrlist1,ArrayList<Integer> arrlist2)
	{
	TreeSet<Integer> ts1=new TreeSet<Integer>();
	TreeSet<Integer> ts2=new TreeSet<Integer>();
	TreeSet<Integer> ts3=new TreeSet<Integer>();
	ArrayList<Integer> aa=new ArrayList<Integer>();
	for(int i=0;i<arrlist1.size();i++)
	ts1.add(arrlist1.get(i));

	for(int i=0;i<arrlist2.size();i++)
		ts2.add(arrlist2.get(i));

	ts1.addAll(ts2);

	for(int i=0;i<arrlist1.size();i++)
	{
	for(int j=0;j<arrlist2.size();j++)
	{
	if(arrlist1.get(i)==arrlist2.get(j))
	ts3.add(arrlist1.get(i));
	}
	}
	ts1.removeAll(ts3);
	aa.addAll(ts1);
	int res[]=newint[aa.size()];
	for(int i=0;i<res.length;i++)
	res[i]=aa.get(i);
	return res;


	}
}
```

60. Price Calculator - II

Write a small price calculator application with the below mentioned flow:

1. Read a value n indicating the total count of devices. This would be followed by the name and price of the device. The datatype for name would be String and price would be float.

2. Build a hashmap containing the peripheral devices with name as key and price as value.

3. Read a value m indicating the number of devices for which the price has to be calculated. This would be followed by device names.

4. For each devices mentioned in the array calcuate the total price.

5. You decide to write a function costEstimator which takes the above hashmap and array as input and returns the total price (float) as output with two decimal points. Include this function in class UserMainCode.

Create a Class Main which would be used to read details in step 1 and build the hashmap. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of device details. The first number indicates the size of the devices. The next two values indicate the name,price.

This would be followed by m indicating the size of the device array. The next m values would be the device names.
Output consists of the total price in float.
Refer sample output for formatting specifications.

Sample Input 1:
3
Monitor
1200.36
Mouse
100.42
Speakers

500.25
2
Speakers
Mouse
Sample Output 1:
600.67

```
import java.util.*;

public class UserMainCode {

public static void main(String[] args) {

HashMap<String, String> m1=new HashMap<String, String>();

m1.put("monitor", "1200.36");

m1.put("mouse","100.42");

m1.put("speaker", "500.25");

String[] s={"speaker","mouse"};

System.out.println(getTheTotalCostOfPheripherals(m1,s));

}

public static float getTheTotalCostOfPheripherals(HashMap<String,String> m1,String[] s) {

Float f=(float) 0;

Iterator<String> i=m1.keySet().iterator();

while(i.hasNext()){

String s1=(String) i.next();

Float f1=Float.parseFloat(m1.get(s1));

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

```
if(s[j].equals(s1))

f+=f1;

}

return f;

}}
```

61.String Processing - ZigZag

Write a program to read a string containing date in DD-MM-YYYY format. find the number of days in the given month.

Note - In leap year February has got 29 days.

Include a class UserMainCode with a static method **getLastDayOfMonth** which accepts the string. The return type is the integer having number of days.

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 integer.
Refer sample output for formatting specifications.

Sample Input 1:
12-06-2012
Sample Output 1:
30

Sample Input 2:
10-02-2012
Sample Output 2:
29

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class User {



	public static void main(String[] args) throws IOException, ParseException  {
		// TODO Auto-generated method stub
	String s1="10-02-2012";
	SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
	Calendar cal=Calendar.getInstance();
	Date d1=sdf.parse(s1);
	cal.setTime(d1);
	int n=cal.getActualMaximum(Calendar.DAY_OF_MONTH);
	System.out.println(n);
	}
}
```

62. Leap Year

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.

Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The return type is the boolean indicating 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:

23/02/2012

Sample Output 1:

TRUE

Sample Input 2:

12/12/2011

Sample Output 2:

FALSE

```
            import java.text.ParseException;

import java.util.*;
publicclass Main
{
      publicstaticvoid main(String[] args) throws ParseException  {

                        Scanner sc=new Scanner(System.in);
                        String s=sc.nextLine();
                        System.out.println(User.leapYear(s));

                        }
            }
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

publicclass User
{
      publicstaticboolean leapYear(String s) throws ParseException
      {
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
      GregorianCalendar g=new GregorianCalendar();

      Calendar cal=Calendar.getInstance();
      Date d1=sdf.parse(s);
      cal.setTime(d1);
      intn=cal.get(Calendar.YEAR);

      boolean b=g.isLeapYear(n);
return b;


}}
```

63. Largest Chunk

Write a program to read a string and return the length of the largest "chunk" in the string.
A chunk is a repetition of same character 2 or more number of times. If the given string doest not contain any repeated chunk of characters return -1.
Include a class UserMainCode with a static method **getLargestSpan** which accepts the string. The return type is the integer.
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 integer.
Refer sample output for formatting specifications.

Sample Input 1:
This place is soooo good
Sample Output 1:
4

```
import java.util.*;

public class Main {

public static void main(String[] args) {

String s1="You are toooo good";

System.out.println(maxChunk(s1));

}

public static int maxChunk(String s1) {

int max=0;

StringTokenizer t=new StringTokenizer(s1," ");

while(t.hasMoreTokens()){

String s2=t.nextToken();

int n=0;

for(int i=0;i<s2.length()-1;i++)

if(s2.charAt(i)==s2.charAt(i+1))

n++;

if(n>max)
```

```
max=n;
}
return (max+1);
}
}
```

64. Largest Span

Write a program to read a integer array, find the largest span in the array.
Span is the count of all the elements between two repeating elements including the repeated elements.
Include a class UserMainCode with a static method **getLargestSpan** which accepts the integer array. The return type is integer.

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

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of integer.
Refer sample output for formatting specifications.

Sample Input 1:
6
4
2
1
4
5
7
Sample Output 1:
4

```
public class Main {
public static void main(String[] args) {
int[]a={1,2,1,1,3};
System.out.println(maxSpan(a));
}
public static int maxSpan(int[] a) {
String s2 = null;
int n=0;
StringBuffer sb=new StringBuffer();
for(int i=0;i<a.length;i++)
sb.append(String.valueOf(a[i]));
String s1=sb.toString();
for(int i=0;i<s1.length();i++)
for(int j=i+1;j<s1.length();j++)
if(s1.charAt(i)==s1.charAt(j))
s2=String.valueOf(s1.charAt(j));
int n1=s1.indexOf(s2);
int n2=s1.lastIndexOf(s2);
for(int i=n1+1;i<n2;i++)
n++;
return (n+2);
}
}
```

65.Even Sum & Duplicate Elements

Write a program to read a integer array, Remove the duplicate elements and display sum of even numbers in the output. If input array contain only odd number then return -1.
Include a class UserMainCode with a static method **sumElements** which accepts the integer array. The return type is integer.

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

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of integer.
Refer sample output for formatting specifications.

Sample Input 1:
7
2
3
54
1
6
7
7
Sample Output 1:
62

Sample Input 2:
6
3
7
9
13
17
21
Sample Output 2:
-1

```
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.LinkedHashSet;
import java.util.Scanner;
public class Main
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int a[]=new int[n];

for(int i=0;i<n;i++)
{
	a[i]=sc.nextInt();
}
System.out.println(UserMainCode.display(a));
}}


import java.util.Iterator;
import java.util.LinkedHashSet;

public class UserMainCode {
	public static int display(int a[])
	{
		LinkedHashSet<Integer>h1=new LinkedHashSet<Integer>();
		int s=0;
		for(int i=0;i<a.length;i++)
		{
			h1.add(a[i]);
		}
		Iterator<Integer> it=h1.iterator();
		while(it.hasNext())
		{
			int k=it.next();
			if(k%2==0)
			{
```

```
                    s=s+k;
            }
    }
    if(s>0)
            return s;
    else
            return -1;
    }}
```

66.Regular Expression - III

Given a string (s) apply the following rules.
I)At least 8 characters must be present
II)At least one capital letter must be present
III)At least one small letter must be present
Iv)At least one special symbol must be present
V)At least one numeric value must be present
If the condition is satisifed then print valid else print invalid.

Include a class UserMainCode with a static method **passwordValidation** which accepts the string. The return type is the 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 (valid / invalid) .
Refer sample output for formatting specifications.

Sample Input 1:
Technology$1213
Sample Output 1:
valid

```
public class UserMainCode

{

    public static int display(String s)

    {



        if(s.matches(".*[0-9]{1,}.*") && s.matches(".*[@#$]{1,}.*") && s.length()>=8 &&
s.matches(".*[A-Z]{1,}.*") && s.matches(".*[a-z]{1,}.*"))

            return 1;

        else

            return -1;

    }}
```

```
import java.util.*;
publicclass Main
{
      publicstaticvoid main(String[] args)      {

                        Scanner sc=new Scanner(System.in);
                        String s=sc.nextLine();
                        System.out.println(User.leapYear(s));

                        }
            }

publicclass User{
publicstaticint leapYear(String s)
      {

if(s.matches(".*[0-9]{1,}.*")
            &&s.matches(".*[!@#$%^&*]{1,}.*") &&s.length()>=8 &&
            s.matches(".*[A-Z]{1,}.*") &&s.matches(".*[a-z]{1,}.*"))
return 1;
else
return -1;
      }}
```

67.Integer Factorial

Give an array of integer as input, store the numbers and their factorials in an hashmap and print the same.
Include a class UserMainCode with a static method **getFactorial** which accepts the integer array. The return type is the hashmap which is printed key:value.

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

Input and Output Format:
Input consists of a number denoting the size of the array and followed by the elements.
Output consists of a hashmap printed in the output format .
Refer sample output for formatting specifications.

Sample Input1:
4
2
3
5
4
Sample Output1:
2:2
3:6
5:120
4:24

```
import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;


import java.util.Scanner;
```

```
public class kapes3 {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());

int []a=new int[s];

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

{

a[i]=sc.nextInt();

}

LinkedHashMap<Integer,Integer>hm2=new LinkedHashMap<Integer,Integer>();

hm2=kapes4.display(a);

Iterator<Integer> it=hm2.keySet().iterator();

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

{

int n=it.next();

int fac=hm2.get(n);

System.out.println(n+":"+fac);

}

}

}


import java.text.DecimalFormat;

import java.util.HashMap;
```

```
import java.util.Iterator;

import java.util.LinkedHashMap;

public class kapes4

{public static LinkedHashMap<Integer,Integer> display(int[] a)

{


LinkedHashMap<Integer,Integer>hm=new LinkedHashMap<Integer,Integer>();

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

{

int u=1;

for(int j=1;j<=a[i];j++)

{

u=u*j;

}

hm.put(a[i],u);

}

return hm;

}}
```

68. String processing – Long + Short + Long

Obtain two strings S1,S2 from user as input. Your program should form a string of "long+short+long", with the shorter string inside of the longer String.
Include a class UserMainCode with a static method **getCombo** which accepts two string variables. The return type is the string.

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

Input and Output Format:
Input consists of two strings with maximum size of 100 characters.
Output consists of an string.

Refer sample output for formatting specifications.
Sample Input 1:
Hello
Hi
Sample Output 1:
HelloHiHello

```
import java.util.StringTokenizer;


public class User {

public static void main(String[] args){

	String s1="Hi";

	String s2="Hello";

	System.out.println(capsStart(s1,s2));

}

public static String capsStart(String s1,String s2){

	StringBuffer s5=new StringBuffer();

	int q=s1.length();

	int w=s2.length();

	if(q>w)

	{

		s5.append(s1).append(s2).append(s1);
```

```
        }
        else
        {
                s5.append(s2).append(s1).append(s2);
        }
        return s5.toString();
}
}
```

69. Age for Voting

Given a date of birth (dd/MM/yyyy) of a person in string, compute his age as of 01/01/2015.

If his age is greater than 18, then println eligible else println not-eligible.

Include a class UserMainCode with a static method getAge which accepts the string value. The return type is the string.

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

Input and Output Format:
Input consists of two string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
16/11/1991

Sample Output 1:

eligible

```
import java.util.*;
publicclass Main
{
	publicstaticvoid main(String[] args)	{

		Scanner sc=new Scanner(System.in);
		String s =sc.nextLine();
		System.out.println(User.display(s));
		}}
```

```
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
publicclass User{


	publicstatic String display(String n)
	{

	int year=0;
	 String now="01/01/2015";
	SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yyyy");
	try
	{

	sdf1.setLenient(false);
	Calendar c1=Calendar.getInstance();
	Date d=sdf1.parse(n);
	c1.setTime(d);
	int y=c1.get(Calendar.YEAR);
	int m=c1.get(Calendar.MONTH);
	int day=c1.get(Calendar.DAY_OF_MONTH);


	Calendar c2=Calendar.getInstance();
	Date d1=sdf1.parse(now);
	c1.setTime(d1);
	int y1=c2.get(Calendar.YEAR);
	int m1=c2.get(Calendar.MONTH);
	int day1=c2.get(Calendar.DAY_OF_MONTH);

	year=y1-y;
	//System.out.println(year);
	if(m>m1)
	year--;
	elseif(m==m1)
```

```
{if(day<day1)
year--;
}
}
catch(Exception e)
{
e.printStackTrace();
}
if(year>18)
return"eligible";
else
return"not-eligible";
}}
```

```
public class UserMainCode{
    public static int getMaxSpan(int a[]) {
        int i,j,k,count,max=0,p=0;
        int n=a.length;
        for(i=0;i<n;i++)
        {
            count=0;
            for(j=i+1;j<n;j++)
            {
                if(a[i]==a[j])
                {
                    p=j;
                }
            }
            for(k=i;k<=p;k++)
            {
                count++;
            }
            if(count>max)
            {
                max=count;
            }
```

```
        }
        return max;
        }


}
```

1. Unique Even Sum

Write a program to read an array, eliminiate duplicate elements and calculate the sum of even numbers (values) present in the array.

Include a class UserMainCode with a static method **addUniqueEven** which accepts a single integer array. The return type (integer) should be the sum of the even numbers. In case there is no even number it should return -1.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

In case there is no even integer in the input array, print **no even numbers** as output. Else print the sum.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

4

2

5

1

4

Sample Output 1:

6

Sample Input 2:

3

1

1

1

Sample Output 2:

no even numbers

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = newint[20];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
int res = User.addUniqueEven(a);
if (res == -1)
System.out.println("no even numbers");
else
System.out.println(res);

}
}


publicclass User {
publicstaticint addUniqueEven(int a[]) {
int i = 0, j = 0, count = 0, sum = 0;
int n = a.length;
for (i = 0; i < n; i++) {
count = 0;
for (j = i + 1; j < n; j++) {
if (a[i] == a[j])
count++;
}
if (count == 0) {
if (a[i] % 2 == 0)
sum = sum + a[i];
}
}
if (sum == 0)
```

```
return -1;
else
returnsum;
}
}
```

2. Palindrome & Vowels

Write a program to check if a given string is palindrome and contains at least two different vowels.

Include a class UserMainCode with a static method **checkPalindrome** which accepts a string. The return type (integer) should be 1 if the above condition is satisfied, otherwise return -1.

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

Note – Case Insensitive while considering vowel, i.e a &A are same vowel, But Case sensitive while considering palindrome i.e abc CbA are not palindromes.

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

abceecba

Sample Output 1:

valid

Sample Input 2:

abcd

Sample Output 2:

Invalid

Solution :

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
int res = User.checkPalindrome(s);
if (res == 1)
System.out.println("valid");
else
System.out.println("invalid");

}
}


publicclass User {
publicstaticint checkPalindrome(String s) {
int res = 0, i = 0, j = 0, count = 0, k = 0;
StringBuffer sb = new StringBuffer(s);
sb.reverse();
if (sb.toString().equals(s)) {
for (i = 0; i < s.length(); i++) {
count = 0;

for (j = i + 1; j < s.length(); j++) {
if (s.charAt(i) == s.charAt(j))
count++;
}
if (count == 0)
if (s.charAt(i) == 'a' || s.charAt(i) == 'e'
|| s.charAt(i) == 'i' || s.charAt(i) == 'o'
|| s.charAt(i) == 'u' || s.charAt(i) == 'A'
|| s.charAt(i) == 'E' || s.charAt(i) == 'I'
|| s.charAt(i) == 'O' || s.charAt(i) == 'U')
k++;
}
}
if (k >= 2)
res = 1;
else
res = 0;

return res;
}
```

}

3. Strings – Unique & Existing Characters

Obtain two strings from user as input. Your program should modify the first string such that all the characters are replaced by plus sign (+) except the characters which are present in the second string.

That is, if one or more characters of first string appear in second string, they will not be replaced by +.

Return the modified string as output. Note - ignore case.

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

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

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

abcxyz

axdef

Sample Output 1:

a++ x++

Sample Input 2:

ABCDEF

feCBAd

Sample Output 2:

ABCDEF

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s1 = sc.nextLine();
String s2 = sc.nextLine();
System.out.println(User.replacePlus(s1, s2));
}
}

publicclass User {
publicstatic String replacePlus(String s1, String s2) {
String ss1 = s1.toLowerCase();
String ss2 = s2.toLowerCase();
StringBuffer sb = newStringBuffer();
for (int i = 0; i < s1.length(); i++) {
char c = ss1.charAt(i);
if (ss2.indexOf(c) == -1)
sb.append('+');
else
sb.append(s1.charAt(i));
}
return sb.toString();
}
}
```

4. Longest Word

Write a Program which finds the longest word from a sentence. Your program should read a sentence as input from user and return the longest word. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method **getLargestWord** which accepts a string The return type is the longest word of type string.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

Welcome to the world of Programming

Sample Output 1:

Programming

Sample Input 2:

ABC DEF

Sample Output 2:

ABC

Solution:

```
import java.util.Scanner;

publicclass Main {

	publicstaticvoid main(String[] args) {
		Scanner s = newScanner(System.in);
		String s1 = s.nextLine();
		System.out.println(User.getLongestWord(s1));
	}

}

publicclass User {
	publicstatic String getLongestWord(String s) {
```

```
            int len, i, p = 0, max = 0, count = 0;
            char b;
            s = s.concat(" ");
            len = s.length();
            for (i = 0; i < len; i++) {
                  b = s.charAt(i);
                  if (b != ' ') {
                        count++;
                  } else {
                        if (count > max) {
                              max = count;
                              p = i;
                        }
                        count = 0;
                  }
            }
            return (s.substring(p - max, p));
      }
}
```

```
import java.util.Scanner;
public class PalindromeMain {

      public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();

            System.out.println(Palindrome.checkPalindrome(s1));
                  }

}
```

```
import java.util.StringTokenizer;


public class Palindrome {

      public static String checkPalindrome(String s1)
      {
    int res,max=0;
    String s2=null;
      StringTokenizer st=new StringTokenizer(s1," ");
      while(st.hasMoreTokens())
      {
            String s=st.nextToken();
            res=s.length();
            if(res>max)
            {
```

```
                max=res;

        s2=s;

        }
}
        return s2;

}
        }
```

5. String Occurences

Obtain two strings from user as input. Your program should count the number of occurences of second word of second sentence in the first sentence.

Return the count as output. Note - Consider case.

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

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

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

abc bcd abc bcd abc abc

av abc

Sample Output 1:

4

Sample Input 2:

ABC xyz AAA

w abc

Sample Output 2:

0

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String s1 = s.nextLine();
String s2 = s.nextLine();
System.out.println(User.countNoOfWords(s1, s2));
}
}


import java.util.StringTokenizer;
publicclass User {
publicstaticint countNoOfWords(String s1, String s2) {
String[] a = new String[s1.length()];
String[] b = new String[s2.length()];
int i = 0, j = 0, count = 0;
StringTokenizer st1 = newStringTokenizer(s1, " ");
StringTokenizer st2 = newStringTokenizer(s2, " ");

while (st1.hasMoreTokens()) {
a[i] = st1.nextToken();
i++;
}
while (st2.hasMoreTokens()) {
b[j] = st2.nextToken();
j++;
}
for (int k = 0; k < i; k++) {
if (b[1].equals(a[k])) {
count++;
}
}
return count;
}
}
```

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

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();
            String s2 = sc.nextLine();
            System.out.println(Palindrome.checkPalindrome(s1,s2));
                  }

}
```

```
import java.util.StringTokenizer;


public class Palindrome {

      public static int checkPalindrome(String s1,String s2)
      {

    int count=0;
      StringTokenizer st=new StringTokenizer(s1," ");
      StringTokenizer st1=new StringTokenizer(s2," ");
      String a2=st1.nextToken();
      String b2=st1.nextToken();

      while(st.hasMoreTokens())
      {
            String s=st.nextToken();

            if(s.equalsIgnoreCase(b2))
            {
                  count++;

            }
      }
            return count;

      }
            }
```

6. ArrayList Manipulation

Write a program that performs the following actions:

1. Read 2n integers as input.
2. Create two arraylists to store n elements in each arraylist.
3. Write a function **generateOddEvenList** which accepts these two arraylist as input.
4. The function fetch the odd index elements from first array list and even index elements from second array list and add them to a new array list according to their index.
5. Return the arraylist.

Include a class UserMainCode with the static method **generateOddEvenList** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read 2n integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

- Consider 0 as an even number.

- Maintain order in the output array list

Input and Output Format:

Input consists of 2n+1 integers. The first integer denotes the size of the arraylist, the next n integers are values to the first arraylist, and the last n integers are values to the second arraylist.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

Sample Input 1:

5

12

13

14

15

16

2

3

4

5

6

Sample Output 1:

2

13

4

15

6

Solution :

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

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
ArrayList<Integer> al1 = new ArrayList<Integer>();
ArrayList<Integer> al2 = new ArrayList<Integer>();
ArrayList<Integer> a = new ArrayList<Integer>();
for (int i = 0; i < n; i++)
al1.add(s.nextInt());
for (int i = 0; i < n; i++)
al2.add(s.nextInt());
a = User.generateOddEvenList(al1, al2);
for (inti = 0; i< a.size(); i++)
System.out.println(a.get(i));
}
}
```

```
import java.util.ArrayList;

publicclass User {
publicstatic ArrayList<Integer> generateOddEvenList(ArrayList<Integer> a1,
ArrayList<Integer> a2)
```

```
{
ArrayList<Integer> a = new ArrayList<Integer>();
int i = 0;
for (i = 0; i < a1.size(); i++) {
if (i % 2 == 0)
a.add(a2.get(i));
else
a.add(a1.get(i));
}
return a;

}
}
```

7. Duplicate Characters

Write a Program which removes duplicate characters from the string. Your program should read a sentence (string) as input from user and return a string removing duplicate characters. Retain the first occurance of the duplicate character. Assume the characters are case – sensitive.

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

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

hi this is sample test

Sample Output 1:

hi tsample

Sample Input 2:

ABC DEF

Sample Output 2:

ABC DEF

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String ss = s.nextLine();
System.out.println(User.removeDuplicates(ss));
}
}


import java.util.Iterator;
importjava.util.LinkedHashSet;

publicclass User {
publicstatic String removeDuplicates(String s) {
char a[] = s.toCharArray();
StringBuffer sb = newStringBuffer();
LinkedHashSet<Character> lh = newLinkedHashSet<Character>();
for (int i = 0; i < a.length; i++)
lh.add(a[i]);
Iterator<Character> itr = lh.iterator();
while (itr.hasNext()) {
char c = itr.next();
if (c != ' ')
;
sb.append(c);
}
return sb.toString();
}
}
```

```
import java.util.Scanner;
class Main
{	public static void main(String[] arg)
	{
		Scanner sc=new Scanner(System.in);
		String s=sc.nextLine();
	System.out.println(MainClass.removeDuplicate(s));
	}}



import java.util.HashSet;
import java.util.LinkedHashSet;
public class MainClass {

	public static  String removeDuplicate(String s)
	{
	  LinkedHashSet<Character> has=new LinkedHashSet<Character>();
		for(int i=0;i<s.length();i++)
		{
			has.add(s.charAt(i));
		}
		StringBuffer sb=new StringBuffer();
		for(Character c:has)
		{
			sb.append(c);
		}
		return sb.toString();
}
}
```

8. Mastering Hashmap

You have recently learnt about hashmaps and in order to master it, you try and use it in all of your programs.

Your trainer / teacher has given you the following exercise:

1. Read 2n numbers as input where the first number represents a key and second one as value. Both the numbers are of type integers.
2. Write a function **getAverageOfOdd** to find out average of all values whose keys are represented by odd numbers. Assume the average is an int and never a decimal number. Return the average as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read 2n numbers and build the hashmap. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of a 2n+ 1 integers. The first integer specifies the value of n (essentially the hashmap size). The next pair of n numbers denote the key and value.

Output consists of an integer representing the average.

Refer sample output for formatting specifications.

Sample Input 1:

4

2

34

1

4

5

12

4

22

Sample Output 1:

8

Solution:

```
import java.util.HashMap;
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
HashMap<Integer, Integer> hm1 = new HashMap<Integer, Integer>();
for (int i = 0; i < n; i++)
hm1.put(s.nextInt(), s.nextInt());
System.out.println(User.getAverageOfOdd(hm1));
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;

publicclass User {
publicstaticint getAverageOfOdd(HashMap<Integer, Integer> hm1) {
int sum = 0, count = 0;
Iterator<Integer> itr = hm1.keySet().iterator();
while (itr.hasNext()) {
int key = itr.next();
if (key % 2 != 0) {
count++;
int val = hm1.get(key);
sum = sum + val;
}
}
int avg = sum / count;
return avg;
}
}
```

9. **Managers & Hashmaps**

A Company wants to automate its payroll process. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, designation and salary in the given order. The datatype for id is integer, designation is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and designation as value, and the second hashmap contains same employee ids as key and salary as value.
3. The company decides to hike the salary of managers by 5000. You decide to write a function **increaseSalaries** which takes the above hashmaps as input and returns a hashmap with only managers id and their increased salary as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee designation and employee salary.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

2

2

programmer

3000

8

manager

50000

Sample Output 1:

8

55000

Solution :

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

class Main {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
HashMap<Integer, String> h1 = new HashMap<Integer, String>();
HashMap<Integer, Integer> h2 = new HashMap<Integer, Integer>();
HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
for (int i = 0; i < n; i++) {
int id = sc.nextInt();
h1.put(id, sc.next());
h2.put(id, sc.nextInt());
}
hm = User.dis(n, h1, h2);
Iterator<Integer> itr = hm.keySet().iterator();
while (itr.hasNext()) {
int id = itr.next();
int sal = hm.get(id);
System.out.println(id);
System.out.println(sal);
}
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;

public class User {
public static HashMap<Integer, Integer> dis(int n,
HashMap<Integer, String> h1, HashMap<Integer, Integer> h2) {
HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

Iterator<Integer> itr = h1.keySet().iterator();
while (itr.hasNext()) {
int id = itr.next();
String deg = h1.get(id);
```

```
if (deg.equalsIgnoreCase("manager")) {
hm.put(id, h2.get(id) + 5000);
}
}
return hm;

}
}
```

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

      Scanner sc=new Scanner(System.in);

      HashMap<Integer,String> ip1=new  HashMap<Integer,String>();
      HashMap<Integer,Integer> ip2=new  HashMap<Integer,Integer>();
      int n=Integer.parseInt(sc.nextLine());

      for(int i=0;i<n;i++)
      {
            int id=Integer.parseInt(sc.nextLine());


      ip1.put(id,sc.nextLine());

      ip2.put(id,Integer.parseInt(sc.nextLine()));
      }

      HashMap<Integer,Integer> op=new  HashMap<Integer,Integer>();
      op=MainClass.addsal(ip1,ip2);

      Iterator<Integer> itr=op.keySet().iterator();
      while(itr.hasNext())
      {
            int  key=itr.next();
```

```
            int value=op.get(key);
            System.out.println(key);
            System.out.println(value);

      }


}}

/*

int n=sc.nextInt();
      for(int i=0;i<n;i++)
      {
      int id=sc.nextInt();
      ip1.put(id,sc.next());
      ip2.put(id,sc.nextInt());
      }

*/


import java.util.HashMap;
import java.util.Iterator;

public class MainClass {
      public static HashMap<Integer,Integer>
addsal(HashMap<Integer,String> hm1,
                  HashMap<Integer,Integer> hm2)
                  {
             HashMap<Integer,Integer>op=new
HashMap<Integer,Integer>();

             Iterator<Integer> itr=hm1.keySet().iterator();

             while(itr.hasNext())
             {
                   int id=itr.next();
                   String s=hm1.get(id);
                   if(s.equals("manager"))
                   {
                         int newsal=hm2.get(id)+5000;
                         op.put(id,newsal);
                   }

             }
             return op;

                        }
}
```

10. Check first and last word

Write a program to check if the first word and the last word in the input string match.

Include a class **UserMainCode** with a static method **"check"** that accepts a string argument and returns an int. If the first word and the last word in the string match, the method returns the number of characters in the word. Else the method returns the sum of the number of characters in the first word and last word.

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

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

Sample Input 1:
how are you you are how

Sample Output 1:
3

Sample Input 2:
how is your child

Sample Output 2:
8

Solution:

```
import java.util.Scanner;

publicclass Main {
```

```
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String ss = s.nextLine();
System.out.println(User.check(ss));
}
}
```

```
import java.util.StringTokenizer;

publicclass User {
publicstaticint check(String s) {
StringTokenizer st = new StringTokenizer(s, " ");
int n = st.countTokens();
String[] s1 = new String[n];
int i = 0, value = 0;
while (st.hasMoreTokens()) {
s1[i] = st.nextToken();
i++;
}
if (s1[0].equals(s1[i - 1]))
value = s1[0].length();
else
value = s1[0].length() + s1[i - 1].length();
return value;
}
}
```

11. Concatenate Characters

Given an array of Strings, write a program to take the last character of each string and make a new String by concatenating it.

Include a class **UserMainCode** with a static method **"concatCharacter"** that accepts a String array as input and returns the new String.

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

Input and Output Format:
The first line of the input consists of an integer n that corresponds to the number of strings in the input string array.
The next n lines of the input consist of the strings in the input string array.

Output consists of a string.

Sample Input:
3
ab
a
abcd

Sample Output:
bad

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
String[] str = new String[n];
for (int i = 0; i < n; i++)
str[i] = s.next();
System.out.println(User.concatCharacter(str));
}
}


publicclass User {
publicstatic String concatCharacter(String[] s) {

StringBuffer sb = newStringBuffer();
for (int i = 0; i < s.length; i++) {
sb.append(s[i].charAt(s[i].length() - 1));
}
return sb.toString();
}
}
```

12. Anagram

Write a program to check whether the two given strings are anagrams.

Note: Rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once is called Anagram."

Include a class **UserMainCode** with a static method **"getAnagram"** that accepts 2 strings as arguments and returns an int. The method returns 1 if the 2 strings are anagrams. Else it returns -1.

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

Input and Output Format:
Input consists of 2 strings. Assume that all characters in the string are lower case letters.
Output consists of a string that is either "Anagrams" or "Not Anagrams".

Sample Input 1:
eleven plus two
twelve plus one

Sample Output 1:
Anagrams

Sample Input 2:
orchestra
carthorse

Sample Output 2:
Anagrams

Sample Input 3:
cognizant
technologies

Sample Output 3:
Not Anagrams

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String s1 = s.nextLine();
String s2 = s.nextLine();
int result = User.getAnagrams(s1, s2);
```

```
if (result == 1)
System.out.println("Anagrams");
else
System.out.println("Not Anagrams");
}
}
```

```
import java.util.ArrayList;
import java.util.Collections;

publicclass User {
publicstaticint getAnagrams(String s1, String s2) {

String str1 = s1.toLowerCase();
String str2 = s2.toLowerCase();
ArrayList<Character> al1 = new ArrayList<Character>();
ArrayList<Character> al2 = new ArrayList<Character>();
ArrayList<Character> al3 = new ArrayList<Character>();
int res = 0;
for (int i = 0; i < s1.length(); i++)
al1.add(str1.charAt(i));
for (int i = 0; i < s2.length(); i++)
al2.add(str2.charAt(i));
al3.add(' ');
al1.removeAll(al3);
al2.removeAll(al3);
Collections.sort(al1);
Collections.sort(al2);
if (al1.equals(al2))
res = 1;
else
res = -1;
return res;
}
}
```

```
public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();
            String s2 = sc.nextLine();
            boolean b =Anagrams.check(s1, s2);
            if (b == true)
            System.out.println("TRUE");
            else
            System.out.println("FALSE");
```

```
	}
}
```

```
	public class Anagrams
	{
		public static boolean check(String s1,String s2)
		{
		boolean res=false;
			ArrayList<Character> a1=new ArrayList<Character>();
			ArrayList<Character> a2=new ArrayList<Character>();
			for(int i=0;i<s1.length();i++)
			{
				a1.add(s1.charAt(i));
			}

			for(int i=0;i<s2.length();i++)
			{
				a2.add(s2.charAt(i));
			}
			Collections.sort(a1);
			Collections.sort(a2);


			if((a1.containsAll(a2))|| (a2.containsAll(a1)))
			{
				res=true;
			}
			return res;
		}
	}
```

13. Calculate Meter Reading

Given 2 strings corresponding to the previous meter reading and the current meter reading, write a program to calculate electricity bill.
The input string is in the format ""AAAAAXXXXX"".
AAAAA is the meter code and XXXXX is the meter reading.
FORMULA: (XXXXX-XXXXX)*4

Hint: if AAAAA of input1 and input2 are equal then separate the XXXXX from string and convert to integer. Assume that AAAAA of the 2 input strings will always be equal.

Include a class **UserMainCode** with a static method **"calculateMeterReading"** that accepts 2 String arguments and returns an integer that corresponds to the electricity bill. The 1st argument corresponds to the previous meter reading and the 2ndarguement corresponds to the current meter reading.

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

Input and Output Format:
Input consists of 2 strings. The first input corresponds to the previous meter reading and the second input corresponds to the current meter reading.

Output consists of an integer that corresponds to the electricity bill.

Sample Input:
CSECE12390
CSECE12400

Sample Output:
40

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String s1 = s.nextLine();
String s2 = s.nextLine();
System.out.println(User.calculateMeterReading(s1, s2));
}
}


publicclass User {
publicstaticint calculateMeterReading(String s1, String s2) {
String str1 = s1.substring(s1.length() / 2);
String str2 = s2.substring(s2.length() / 2);
int a = Integer.parseInt(str1);
int b = Integer.parseInt(str2);
```

```
int res = (b - a) * 4;
return res;
}
}
```

14. Retirement

Given an input as HashMap which contains key as the ID and dob as value of employees, write a program to find out employees eligible for retirement. A person is eligible for retirement if his age is greater than or equal to 60.

Assume that the current date is 01/01/2014.

Include a class **UserMainCode** with a static method “retirementEmployeeList” that accepts a HashMap<String,String> as input and returns a ArrayList<String>. In this method, add the Employee IDs of all the retirement eligible persons to list and return the sorted list.
(Assume date is in dd/MM/yyyy format).

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

Input and Output Format:
The first line of the input consists of an integer n, that corresponds to the number of employees.
The next 2 lines of the input consists of strings that correspond to the id and dob of employee 1.
The next 2 lines of the input consists of strings that correspond to the id and dob of employee 2.
and so on...
Output consists of the list of employee ids eligible for retirement in sorted order.

Sample Input :
4
C1010
02/11/1987
C2020
15/02/1980
C3030
14/12/1952
T4040
20/02/1950

Sample Output:

[C3030, T4040]

Solution:

```
import java.text.ParseException;
importjava.util.LinkedHashMap;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner s = newScanner(System.in);
int n = s.nextInt();
LinkedHashMap<String, String> hm = newLinkedHashMap<String, String>();
for (int i = 0; i < n; i++)
hm.put(s.next(), s.next());
System.out.println(User.retirementEmployeeList(hm));
}
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;

publicclass User {
publicstaticArrayList<String> retirementEmployeeList(
LinkedHashMap<String, String> hm) throws ParseException {
ArrayList<String> al = new ArrayList<String>();
SimpleDateFormat sdf = newSimpleDateFormat("dd/MM/yyyy");
String s = "01/01/2014";
Date d2 = sdf.parse(s);
Date d1 = newDate();
Iterator<String> itr = hm.keySet().iterator();
while (itr.hasNext()) {
String key = itr.next();
String val = hm.get(key);
d1 = sdf.parse(val);
Calendar c = Calendar.getInstance();
c.setTime(d1);
int y1 = c.get(Calendar.YEAR);
int m1 = c.get(Calendar.MONTH);
int day1 = c.get(Calendar.DAY_OF_MONTH);
c.setTime(d2);
```

```
int y2 = c.get(Calendar.YEAR);
int m2 = c.get(Calendar.MONTH);
int day2 = c.get(Calendar.DAY_OF_MONTH);
int y = Math.abs(y1 - y2);
if (m1 == m2) {
if (day1 > day2)
y--;
} elseif (m1 > m2)
y--;
if (y >= 60)
al.add(key);
}
return al;
}
}
```

```
import java.text.ParseException;
import java.util.HashMap;
import java.util.Scanner;

public class NewClassMain {

	public static void main(String[] args) throws ParseException {


		Scanner sc=new Scanner(System.in);
		int n=sc.nextInt();
		int i=0;
		sc.nextLine();
		HashMap<String,String> hm=new HashMap<String,String>();
		for(i=0;i<n;i++)
		{
			hm.put(sc.nextLine(),sc.nextLine());
		}

		System.out.println(NewClass.retirement(hm));

			}

}


public class NewClass {
```

```
	public static ArrayList<String> retirement(HashMap<String,String> hm)
throws ParseException
	{
		ArrayList<String> a1=new ArrayList<String>();
	String s="01/01/2014";
	Iterator<String> itr=hm.keySet().iterator();
	while(itr.hasNext())
	{
	String k=itr.next();
	String dob=hm.get(k);

	SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
	Date d1=sdf.parse(dob);
	Date d2=sdf.parse(s);

	Calendar cal=Calendar.getInstance();
	cal.setTime(d1);

	int y1= cal.get(Calendar.YEAR);
		cal.setTime(d2);

	int y2= cal.get(Calendar.YEAR);

	int res=y2-y1;
	if(res>=60)

	{
		a1.add(k);
	}

	}

	Collections.sort(a1);

	return a1;

	}


	}
```

15. Kaprekar Number

Write a program to check whether the given input number is a Kaprekar number or not.
Note : A positive whole number ‘n’ that has ‘d’ number of digits is squared and split into two pieces, a right-hand piece that has ‘d’ digits and a left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of the two pieces is equal to the number, then ‘n’ is a Kaprekar number.

If its Kaprekar number assign to output variable 1 else -1.
Example 1:
Input1:9
9^2 = 81, right-hand piece of 81 = 1 and left hand piece of 81 = 8
Sum = 1 + 8 = 9, i.e. equal to the number. Hence, 9 is a Kaprekar number.

Example 2:
Input1:45
Hint:
45^2 = 2025, right-hand piece of 2025 = 25 and left hand piece of 2025 = 20
Sum = 25 + 20 = 45, i.e. equal to the number. Hence, 45 is a Kaprekar number."

Include a class **UserMainCode** with a static method “**getKaprekarNumber**” that accepts an integer argument and returns an integer. The method returns 1 if the input integer is a Kaprekar number. Else the method returns -1.

Create a class **Main** which would get the an Integer as input and call the static method **getKaprekarNumber** present in the UserMainCode.

Input and Output Format:
Input consists of an integer.
Output consists of a single string that is either “Kaprekar Number” or “Not A Kaprekar Number”

Sample Input 1:
9

Sample Output 1:
Kaprekar Number

Sample Input 2:
45

Sample Output 2:
Kaprekar Number

Sample Input 3:
4

Sample Output 3:
Not A Kaprekar Number

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int i = User.getKaprekarNumber(n);
if (i == 1)
System.out.println("Kaprekar Number");
else
System.out.println("Not Kaprekar Number");
}
}


public class User {
public static int getKaprekarNumber(int temp) {
int n = temp;
int sq = n * n;
int sqr=sq;
int res = 0;
int count = 0;
while (sq != 0) {
count++;
sq= sq / 10;
}
//String a = Integer.toString(sqr);

String a=String.valueOf(sqr);


String n1 = a.substring(count/2);
String n2 = a.substring(0,count/2);
int i = Integer.parseInt(n1);
int j = Integer.parseInt(n2);
if ((i + j) == temp)
res = 1;
```

```
else
res = -1;
return res;
}
}
```

```
public class Palindrome {
      public static  int  removeDuplicate(int n)
      {
            int temp = n;
            int sq = n * n;
            int sqr=sq;
            int res = 0;
            String sqs=String.valueOf(sq);
            int len=sqs.length();
            String a = String.valueOf(sqr);
            String n1 = a.substring(len/2);
            String n2 = a.substring(0,len/2);
            int i = Integer.parseInt(n1);
            int j = Integer.parseInt(n2);
            if ((i + j) == temp)
            res = 1;
            else
            res = -1;
            return res;

}
}
```

16. Vowels

Given a String input, write a program to find the word which has the the maximum number of vowels. If two or more words have the maximum number of vowels, print the first word.

Include a class **UserMainCode** with a static method "**storeMaxVowelWord**" that accepts a string argument and returns the word containing the maximum number of vowels.

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

Input and Output Format:
Input consists of a string. The string may contain both lower case and upper case letters.
Output consists of a string.

Sample Input :

What is your name?

Sample Output :
Your

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = newScanner(System.in);
String s = sc.nextLine();
System.out.println(User.storeMaxVowelWord(s));
}
}
```

```
import java.util.StringTokenizer;

publicclass User {
publicstatic String storeMaxVowelWord(String s) {
StringTokenizer st = new StringTokenizer(s, " ");
int count = 0, max = 0;
String s2 = null;
while (st.hasMoreTokens()) {
String s1 = st.nextToken();
count = 0;
for (int i = 0; i <s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'e'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'o'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'E' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'O' || s1.charAt(i) == 'U')
count++;
}
if (count > max) {
max = count;
s2 = s1;
}
}
return s2;

}
}
```

17. Unique Characters

Given a String as input , write a program to count and print the number of unique characters in it.

Include a class **UserMainCode** with a static method "**checkUnique**" that accepts a String as input and returns the number of unique characters in it. If there are no unique characters in the string, the method returns -1.

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

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

Sample Input 1:
HOWAREYOU

Sample Output 1:
7

(Hint :Unique characters are : H,W,A,R,E,Y,U and other characters are repeating)

Sample Input 2:
MAMA

Sample Output2:
-1

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
System.out.println(User.checkUnique(s));
}
}
```

```
publicclass User {
publicstaticint checkUnique(String s) {
StringBuffer sb = new StringBuffer(s);
int len = s.length();
int i = 0, j = 0, count;
for (i = 0; i < len; i++) {
count = 0;
for (j = i + 1; j < len; j++) {
if (sb.charAt(i) == sb.charAt(j)) {
sb.deleteCharAt(j);
count++;
j--;
len--;
}
}

if (count > 0) {
sb.deleteCharAt(i);
i--;
len--;
}
}
if(sb.length()==0)
        return -1;
else
return sb.length();
}
}
```

18. Average of Primes

Write a program to read an array and find average of all elements located at index i, where i is a prime number. Type cast the average to an int and return as output. The index starts from 0.

Include a class UserMainCode with a static method **addPrimeIndex** which accepts a single integer array. The return type (integer) should be the average of all elements located at index i where i is a prime number.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20 and minimum number of elements is 3.

Sample Input 1:

4

2

5

2

4

Sample Output 1:

3

Solutions:

```
import java.text.ParseException;
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = newint[20];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
System.out.println(User.addPrimeIndex(a));
}
}

public class User {
public static int addPrimeIndex(int a[],int n) {
int count=0,sum=0,temp=0;
int avg=0;
for(int i=2;i<=n;i++)
```

```
{
count=0;
for(int j=1;j<i;j++)
{
if(i%j==0)
count++;

}
if(count==1)
{
temp++;
sum=sum+a[i];
}
}
avg=sum/temp;
return avg;

}
}
```

19. **ArrayList and Set Operations**

Write a program that performs the following actions:

1. Read 2n integers as input & a set operator (of type char).
2. Create two arraylists to store n elements in each arraylist.
3. Write a function **performSetOperations** which accepts these two arraylist and the set operator as input.
4. The function would perform the following set operations:.

   '+' for SET-UNION

   '*' for SET-INTERSECTION

   '-' for SET-DIFFERENCE

   Refer to sample inputs for more details.
5. Return the resultant arraylist.

Include a class UserMainCode with the static method **performSetOperations** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read 2n+1 integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

Input and Output Format:

Input consists of 2n+2 integers. The first integer denotes the size of the arraylist, the next n integers are values to the first arraylist, and the next n integers are values to the second arraylist and the last input corresponds to that set operation type.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

Sample Input 1:

3

1

2

3

3

5

7

+

Sample Output 1:

1

2

3

5

7

Sample Input 2:

4

10

9

8

7

2

4

6

8

*

Sample Output 2:

8

Sample Input 3:

4

5

10

15

20

0

10

12

20

-

Sample Output 3:

5

15

Solution:

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

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
```

```
ArrayList<Integer> al1 = new ArrayList<Integer>();
ArrayList<Integer> al2 = new ArrayList<Integer>();
ArrayList<Integer> res = new ArrayList<Integer>();
for (int i = 0; i < n; i++)
al1.add(sc.nextInt());
for (int i = 0; i < n; i++)
al2.add(sc.nextInt());
char c = sc.next().charAt(0);
res = User.performSetOperations(al1, al2, c);
for (int i = 0; i < res.size(); i++)
System.out.println(res.get(i));
}
}
```

```
import java.util.ArrayList;
import java.util.LinkedHashSet;

public class User {
public static ArrayList<Integer> performSetOperations(
ArrayList<Integer> al1, ArrayList<Integer> al2, char c) {

LinkedHashSet<Integer> h = new LinkedHashSet<Integer>();
ArrayList<Integer> al3 = new ArrayList<Integer>();
switch (c) {
case '+':
al1.addAll(al2);
h.addAll(al1);
al3.addAll(h);
break;
case '*':
for (int i = 0; i < al1.size(); i++) {
for (int j = 0; j < al2.size(); j++) {
if (al1.get(i) == al2.get(j)) {
al3.add(al1.get(i));
}
}
}
break;
case '-':
for (int i = 0; i < al1.size(); i++) {
for (int j = 0; j < al2.size(); j++) {
if (al1.get(i) == al2.get(j)) {
```

```
al1.remove(i);
}
}
}
al3.addAll(al1);
break;

}

return al3;
}
}
```

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

class Main
{
    public static void main(String[] arg)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        ArrayList<Integer> aa=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            aa.add(sc.nextInt());

        }
        ArrayList<Integer> aa2=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            aa2.add(sc.nextInt());

        }
        char c=sc.next().charAt(0);

        ArrayList<Integer> op=new ArrayList<Integer>();

        op=MainClass.setOPeration(n, aa, aa2, c);
```

```
        Iterator<Integer> itr=op.iterator();
        while(itr.hasNext())
        {
            int a=(Integer)itr.next();
            System.out.println(a);
        }


    }}



import java.util.ArrayList;


public class MainClass {


    public static ArrayList<Integer> setOPeration
    (int n,ArrayList<Integer>aa,ArrayList<Integer>aa2,char c)
    {

        ArrayList<Integer> aa3= new ArrayList<Integer>();

        if(c=='+')
        {
            aa.removeAll(aa2);
            aa.addAll(aa2);
            aa3=aa;
        }


        if(c=='*')
        {
            aa.retainAll(aa2);
            aa3=aa;

            }
    if(c=='-')
    {
        aa.removeAll(aa2);
        aa3=aa;
```

```
    }
return aa3;
}
}
```

20. **Largest Span**

Write a program to read an array and find the size of largest span in the given array

""span"" is the number of elements between two repeated numbers including both numbers. An array with single element has a span of 1.

.

Include a class UserMainCode with a static method **getMaxSpan** which accepts a single integer array. The return type (integer) should be the size of largest span.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

5

1

2

1

1

3

Sample Output 1:

4

Sample Input 2:

7

1

4

2

1

4

1

5

Sample Output 2:

6

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
```

```
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = newint[n];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
System.out.println(User.getLargestSpan(a));

}
}


publicclass User {
publicstaticint getLargestSpan(int[] a) {
int len = a.length;
int i = 0, j = 0, e = 0, count = 0;
for (i = 0; i < len; i++) {
for (j = i + 1; j < len; j++) {
if (a[i] == a[j]) {
e = j;
}
}
if (e - i > count)
count = e - i;
}
return count + 1;
}
}
```

21. max Scorer

Write a program that performs the following actions:

1. Read n strings as input and stores them as an arraylist. The string consists of student information like name and obtained marks of three subjects. Eg: name-mark1-mark2-mark3 [suresh-70-47-12] The mark would range between 0 – 100 (inclusive).
2. Write a function **highestScorer** which accepts these the arraylist and returns the name of the student who has scored the max marks. Assume the result will have only one student with max mark.

Include a class UserMainCode with the static method **highestScorer** which accepts the arraylist and returns the name (string) of max scorer.

Create a Class Main which would be used to read n strings into arraylist and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 1 integer and n strings. The first integer denotes the size of the arraylist, the next n strings are score pattern described above.

Output consists of a string with the name of the top scorer.

Refer sample output for formatting specifications.

Sample Input 1:

3

sunil-56-88-23

bindul-88-70-10

john-70-49-65

Sample Output 1:

john

Solution:

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

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
ArrayList<String> a=new ArrayList<String>();
for(int i=0;i<n;i++)
a.add(sc.next());
System.out.println(User.highestScorer(a));
sc.close();
}
}
```

```
import java.util.ArrayList;
import java.util.StringTokenizer;

public class User {
```

```
public static String highestScorer(ArrayList<String> a) {
String ss=null,name=null,Name=null;
int m1=0,m2=0,m3=0,sum=0,max=0;
for(int i=0;i<a.size();i++)
{
ss=a.get(i);
StringTokenizer st=new StringTokenizer(ss,"-");
while(st.hasMoreTokens())
{
name=st.nextToken();
m1=Integer.parseInt(st.nextToken());
m2=Integer.parseInt(st.nextToken());
m3=Integer.parseInt(st.nextToken());
sum=m1+m2+m3;
if(max<sum)
{
max=sum;
Name=name;
}
}
}
return Name;
}
}
```

22. Max Vowels

Write a Program which fetches the word with maximum number of vowels. Your program should read a sentence as input from user and return the word with max number of vowels. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method **getWordWithMaximumVowels** which accepts a string The return type is the longest word of type string.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

Appreciation is the best way to motivate

Sample Output 1:

Appreciation

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.nextLine();
System.out.println(User.getWordWithMaximumVowels(s));
}
}
```

```
import java.util.StringTokenizer;

publicclass User {
publicstatic String getWordWithMaximumVowels(String s) {
StringTokenizer st = new StringTokenizer(s, " ");
int count = 0, max = 0;
String res = null;
String f = null;
while (st.hasMoreTokens()) {
res = st.nextToken();
count = 0;
for (int k = 0; k < res.length(); k++) {
if (res.charAt(k) == 'a' || res.charAt(k) == 'e'
|| res.charAt(k) == 'i' || res.charAt(k) == 'o'
|| res.charAt(k) == 'u' || res.charAt(k) == 'A'
|| res.charAt(k) == 'E' || res.charAt(k) == 'I'
|| res.charAt(k) == 'O' || res.charAt(k) == 'U')
count++;
if (count > max) {
max = count;
f = res;
}

}
}
```

```
return f;
}
}
```

23. All Vowels

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Return 1 if the condition is satisfied else return -1. Assume there is no repetition of any vowel in the given string and all letters are in lower case.

Include a class UserMainCode with a static method **testOrderVowels** which accepts a string The return type is integer based on the condition stated above.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

acebisouzz

Sample Output 1:

valid

Sample Input 2:

alphabet

Sample Output 2:

invalid

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.nextLine();
int res = User.testOrderVowels(s);
if (res == 1)
```

```
System.out.println("valid");
else
System.out.println("invalid");
}
}
```

```
publicclass User {
publicstaticint testOrderVowels(String s1) {

StringBuffer sb = newStringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}
```

24. Adjacent Swaps

Write a Program that accepts a string as a parameter and returns the string with each pair of adjacent letters reversed. If the string has an odd number of letters, the last letter is unchanged.

Include a class UserMainCode with a static method **swapPairs** which accepts a string. The return type is string which is reversed pair of letters.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

forget

Sample Output 1:

ofgrte

Sample Input 2:

New York

Sample Output 2:

eN woYkr

```
import java.util.Scanner;

public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
String string=sc.nextLine();
System.out.println(User.swapPairs(string));
sc.close();
}
}


public class User {
public static String swapPairs(String s) {
StringBuffer sb=new StringBuffer();
if(s.length()%2==0)
{
for(int i=0;i<s.length()-1;i=i+2)
{
sb.append(s.charAt(i+1)).append(s.charAt(i));
}
}
else
{
for(int i=0;i<s.length()-1;i=i+2)
```

```
{
sb.append(s.charAt(i+1)).append(s.charAt(i));
}
sb.append(s.charAt(s.length()-1));
}
return sb.toString();
}
}
```

25. Sum of Digits

Write a Program that accepts a word as a parameter, extracts the digits within the string and returns its sum.

Include a class UserMainCode with a static method **getdigits** which accepts a string. The return type is integer representing the sum.

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

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

abc12de4

Sample Output 1:

7

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
System.out.println(User.getdigits(s));
}
}
```

```
publicclass User {
publicstaticint getdigits(String s) {
int sum = 0, n = 0;
for (int i = 0; i < s.length(); i++) {
if (s.charAt(i) >= 65 && s.charAt(i) <= 90 || s.charAt(i) >= 97
&&s.charAt(i) <= 122)
;
else {
n = Character.getNumericValue(s.charAt(i));
sum = sum + n;
}
}
return sum;
}
}
```

```
public static String removeDuplicate(String s) {

      int sum = 0,n=0;
      for (int i = 0; i < s.length(); i++) {

            if( Character.isDigit(s.charAt(i))  )
            {
                  int c=Character.getNumericValue(s.charAt(i));
            sum = sum +c;

            }
      }
            String sum1=String.valueOf(sum);
            return sum1;
}
```

26. Password

Given a String , write a program to find whether it is a valid password or not.

Validation Rule:
Atleast 8 characters
Atleast 1 number(1,2,3...)
Atleast 1 special character(@,#,%...)
Atleast 1 alphabet(a,B...)

Include a class **UserMainCode** with a static method "**validatePassword**" that accepts a String argument and returns a boolean value. The method returns true if the password is acceptable. Else the method returns false.
Create a class **Main** which would get a String as input and call the static method **validatePassword** 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:
cts@1010

Sample Output 1:
Valid

Sample Input 2:
punitha3

Sample Output 2:
Invalid

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
boolean flag = User.validatePassword(s);
if (flag == true)
System.out.println("valid");
else
System.out.println("invalid");
}
}

publicclass User {
publicstaticboolean validatePassword(String s) {
int number = 0, c = 0, sp = 0;
boolean flag = false;
for (int i = 0; i < s.length(); i++) {
```

```
if (s.length() >= 8) {
if (Character.isDigit(s.charAt(i))) {
number++;
}
if (Character.isLetter(s.charAt(i))) {
c++;
} else {
if (s.charAt(i) != ''&& !Character.isDigit(s.charAt(i))
&& !Character.isLetter(s.charAt(i)))
sp++;
}
}
}
if (number >= 1 && c >= 1 && sp >= 1)
flag = true;
return flag;
}
}
```

```
public static boolean removeDuplicate(String s) {

                    int number = 0, c = 0, sp = 0,len=0;
                    boolean flag = false;
                    for (int i = 0; i < s.length(); i++)
                    {
                    if (s.length() >= 8)
                    {
                          len++;
                    }
                    if (Character.isDigit(s.charAt(i)))
                    {
                    number++;
                    }
                    if (Character.isLetter(s.charAt(i)))
                    {
                    c++;
                    }

                    if (s.charAt(i) != ' '&& !Character.isDigit(s.charAt(i))
                    && !Character.isLetter(s.charAt(i)))
                    {
                    sp++;
                    }
                    }

                    if (number >= 1 && c >= 1 && sp >= 1 && len>1)
                    flag = true;
```

```
                return flag;

}
```

27. Employee Bonus

A Company wants to give away bonus to its employees. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.
3. If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised salary as value. Assume the age is caculated based on the date 01-09-2014. (Typecast the bonus to integer).
4. Other Rules:

   a. If Salary is less than 5000 store -100.

   b. If the age is less than 25 or greater than 60 store -200.

c. a takes more priority than b i.e both if a and b are true then store -100.

5. You decide to write a function **calculateRevisedSalary** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is “dd-mm-yyyy”

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

2

1010

20-12-1987

10000

2020

01-01-1985

14400

Sample Output 1:

1010

12000

2020

17280

Solution:

```
import java.text.ParseException;
import java.util.*;

public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
TreeMap<Integer,Integer> t=new TreeMap<Integer,Integer>();
HashMap<Integer,String> h1=new HashMap<Integer,String>();
HashMap<Integer,Integer> h2=new HashMap<Integer,Integer>();
for(int i=0;i<n;i++)
{
	int id=sc.nextInt();
	h1.put(id, sc.next());
	h2.put(id, sc.nextInt());
}
t=User.calSalary(h1,h2);
Iterator<Integer> it1=t.keySet().iterator();
```

```
while(it1.hasNext())
{
	int id=it1.next();
	int val=t.get(id);
	System.out.println(id);
	System.out.println(val);
}
sc.close();
}
}


import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class User {
public static TreeMap<Integer,Integer> calSalary(HashMap<Integer,String> h1,
HashMap<Integer,Integer> h2) throws ParseException {
	TreeMap<Integer,Integer> t=new TreeMap<Integer,Integer>();
	Iterator<Integer> it1=h1.keySet().iterator();
	SimpleDateFormat sd=new SimpleDateFormat("dd-MM-yyyy");
	String ss="01-09-2014";
	int new_sal=0;
	while(it1.hasNext())
	{
		int id1=it1.next();
		String dob=h1.get(id1);
		int salary=h2.get(id1);
		Date d1=sd.parse(dob);
		Date d2=sd.parse(ss);
		d1=sd.parse(dob);
		int y1=d1.getYear();
		int y2=d2.getYear();
		int year=Math.abs(y1-y2);
		if(year>=25 && year<=30)
		{
			new_sal=salary+(salary*20/100);
			t.put(id1,new_sal);
		}
		else if(year>=31 && year<=60)
		{
			new_sal=salary+(salary*30/100);
			t.put(id1,new_sal);
		}
		else
			;

	}
	return t;
}
}
```

28. Grade Calculator

A School wants to assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read student details from the User. The details would include roll no, mark in the given order. The datatype for id is integer, mark is integer.
2. You decide to build a hashmap. The hashmap contains roll no as key and mark as value.
3. BUSINESS RULE:

1. If Mark is greater than or equal to 80 store medal as ""GOLD"".

2. If Mark is less then to 80 and greater than or equal to 60 store medal as ""SILVER"".

3 .If Mark is less then to 60 and greater than or equal to 45 store medal as ""BRONZE"" else store ""FAIL"".

Store the result in TreeMap in which Roll No as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of employee details. The first number indicates the size of the students. The next two values indicate the roll id, mark.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

2

1010

80

100

40

Sample Output 1:

100

FAIL

1010

GOLD

Solution:

```
import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;

import java.util.TreeMap;


public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int i;

HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

for (i = 0; i < n; i++) {

hm.put(sc.nextInt(), sc.nextInt());

}

TreeMap<Integer, String> t = new TreeMap<Integer, String>();

t.putAll(User.display(n, hm));

Iterator<Integer> it = t.keySet().iterator();

while (it.hasNext()) {

int r = it.next();
```

```
String g = t.get(r);

System.out.println(r);

System.out.println(g);

}}}

import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;

publicclass User {

publicstatic TreeMap<Integer, String> display(int n,
HashMap<Integer, Integer> h) {
TreeMap<Integer, String> t = new TreeMap<Integer, String>();
Iterator<Integer> i = h.keySet().iterator();
while (i.hasNext()) {
int r = i.next();
int m = h.get(r);
if (m >= 80)
t.put(r, "GOLD");
elseif (m < 80 && m >= 60)
t.put(r, "SILVER");
elseif (m < 60 && m >= 45)
t.put(r, "BRONZE");
else
t.put(r, "FAIL");
}
return t;

}
}
```

29. DigitSum

Write a program to read a non-negative integer n, compute the sum of its digits. If sum is greater than 9 repeat the process and calculate the sum once again until the final sum comes to single digit.Return the single digit.
Include a class UserMainCode with a static method **getDigitSum** which accepts the integer value. The return type is integer.
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:
9999
Sample Output 1:
9

Sample Input 2:
698
Sample Output 2:
5

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
System.out.println(User.getDigitSum(n));
}

}


publicclass User {
publicstaticint getDigitSum(int n) {
int sum = 0;
while (n > 10) {
int r = 0;
sum = 0;
while (n != 0) {
r = n % 10;
sum = sum + r;
n = n / 10;
}
n = sum;
}

return sum;
```

}
}

30. Anagrams

Write a program to read two strings and checks if one is an anagram of the other.
An anagram is a word or a phrase that can be created by rearranging the letters of another given word or phrase. We ignore white spaces and letter case. All letters of 'Desperation' can be rearranged to the phrase 'A Rope Ends It'.
Include a class UserMainCode with a static method **checkAnagram** which accepts the two strings. The return type is boolean which is TRUE / FALSE.
Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two strings.
Output consists of TRUE / FALSE.
Refer sample output for formatting specifications.
Sample Input 1:
tea
eat
Sample Output 1:
TRUE

Sample Input 2:
Desperation
A Rope Ends It
Sample Output 2:
TRUE

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s1 = sc.nextLine();
String s2 = sc.nextLine();
boolean b = User.checkAnagram(s1, s2);
if (b == true)
```

```
System.out.println("TRUE");
else
System.out.println("FALSE");
}
}
```

```
import java.util.ArrayList;
import java.util.Collections;

publicclass User {
publicstaticboolean checkAnagram(String s1, String s2) {
boolean b = false;
ArrayList<Character> a1 = new ArrayList<Character>();
ArrayList<Character> a2 = new ArrayList<Character>();
ArrayList<Character> a3 = new ArrayList<Character>();
for (int i = 0; i < s1.length(); i++)
a1.add(s1.toLowerCase().charAt(i));
for (int i = 0; i < s2.length(); i++)
a2.add(s2.toLowerCase().charAt(i));
a3.add(' ');
a1.removeAll(a3);
a2.removeAll(a3);
Collections.sort(a1);
Collections.sort(a2);
if (a1.equals(a2))
b = true;
return b;
}
}
```

```
public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();
            String s2 = sc.nextLine();
            boolean b =Anagrams.check(s1, s2);
            if (b == true)
            System.out.println("TRUE");
            else
            System.out.println("FALSE");

      }
}
```

```
public class Anagrams
{
	public static boolean check(String s1,String s2)
	{
	boolean res=false;
		ArrayList<Character> a1=new ArrayList<Character>();
		ArrayList<Character> a2=new ArrayList<Character>();
		for(int i=0;i<s1.length();i++)
		{
			a1.add(s1.charAt(i));
		}

		for(int i=0;i<s2.length();i++)
		{
			a2.add(s2.charAt(i));
		}
		Collections.sort(a1);
		Collections.sort(a2);


		if((a1.containsAll(a2))|| (a2.containsAll(a1)))
		{
			res=true;
		}
		return res;
	}
}
```

1. Shift Left

Write a program to read a integer array of scores, and return a version of the given array where all the 5's have been removed. The remaining elements should shift left towards the start of the array as needed,

and the empty spaces at the end of the array should be filled with 0.

So {1, 5, 5, 2} yields {1, 2, 0, 0}.

Include a class UserMainCode with a static method shiftLeft which accepts the integer array. The return type is modified array.

Create a Class Main which would be used to accept the integer array and call the static method

present in UserMainCode.

Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.
Output consists of modified array.
Refer sample output for formatting specifications.

Sample Input 1:

```
7
1
5
2
4
5
3
5
```

Sample Output 1:

```
1
2
4
3
0
0
0
```

Solution:

```
import java.util.Scanner;

public class Main {

	public static void main(String[] args) {
		Scanner s=new Scanner(System.in);
		int n=s.nextInt();
		int a[]=new int[n];
		for(int i=0;i<n;i++)
			a[i]=s.nextInt();
		int res[]=User.shiftLeft(a,n);
		for(int i=0;i<res.length;i++)
			System.out.println(res[i]);

	}

}
```

```
public class User {
	public static int[] shiftLeft(int a[],int n)
	{
		int b[]=new int[n];
		int k=0;
		for(int i=0;i<n;i++)
		{
			if(a[i]!=5)
			{
			b[k]=a[i];
			k++;
		}
		}
		return b;
	}

}
```

32. Word Count

Given a string array (s) with each element in the array containing alphabets or digits. Write a program to add all the digits in every string and return the sum as an integer. If two digits appear simultaneously do not consider it as one number. Ex- For 'Hyderabad 21' consider 2 and 1 as two digits instead of 21 as a number.

Include a class UserMainCode with a static method **sumOfDigits** which accepts the string array. The return type is the integer formed based on rules.
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 an integer indicating the number of elements in the string array.
Output consists of a integer .
Refer sample output for formatting specifications.

Sample Input 1:
5
AAA1
B2B
4CCC
A5
ABCDE
Sample Output 1:

12

Sample Input 2:
3
12
C23
5CR2
Sample Output 2:
15

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
String[] s = new String[n];
for (int i = 0; i < n; i++)
s[i] = sc.next();
System.out.println(User.sumOfDigits(s));

}
}


publicclass User {
publicstaticint sumOfDigits(String[] ss) {
int sum = 0, n = 0;
for (int i = 0; i < ss.length; i++) {
String s = ss[i];
for (int k = 0; k < s.length(); k++) {
if (Character.isDigit(s.charAt(k))) {
n = Character.getNumericValue(s.charAt(k));
sum = sum + n;
}
}
}
return sum;
}
}
```

33. Prefix finder

Given a string array (s) with each element in the array containing 0s and 1s. Write a program to get the number of strings in the array where one String is getting as prefixed in other String in that array .
Example 1: Input: {10,101010,10001,1111} Output =2 (Since 10 is a prefix of 101010 and 10001)
Example 2: Input: {010,1010,01,0111,10,10} Output =3(01 is a prefix of 010 and 0111. Also, 10 is a prefix of 1010) Note: 10 is NOT a prefix for 10.

Include a class UserMainCode with a static method **findPrefix** which accepts the string array. The return type is the integer formed based on rules.
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 an integer indicating the number of elements in the string array followed by the array.
Output consists of a integer .
Refer sample output for formatting specifications.

Sample Input 1:
4
0
1
11
110
Sample Output 1:
3

Solution:

```
import java.util.HashSet;
import java.util.Scanner;


public class Piddi {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
String input[] = new String[size];
for (int i = 0; i < size; i++) {
input[i] = sc.next();
}
```

```
HashSet<String> hs = new HashSet<String>();
for (int i = 0; i < size; i++) {
hs.add(input[i]);
}


size = hs.size();
int i = 0;


int count = 0;
for (i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
if (input[i].equals(input[j]) == false) {
if (input[j].startsWith(input[i])) {
count++;
}
}
}
}
System.out.println(count);
}
}
```

34. Commons

Given two arrays of strings,return the count of strings which is common in both arrays. Duplicate entries are counted only once.
Include a class UserMainCode with a static method **countCommonStrings** which accepts the string arrays. The return type is the integer formed based on rules.
Create a Class Main which would be used to accept the string arrays and integer and call the static method present in UserMainCode.

Input and Output Format:
Input consists of a an integer indicating the number of elements in the string array followed by the array.
Output consists of a integer .
Refer sample output for formatting specifications.

Sample Input 1:
3
a
c
e
3
b
d

e
Sample Output 1:
1

Sample Input 2:
5
ba
ba
black
sheep
wool
5
ba
ba
have
any
wool
Sample Output 2:
2

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n1 = sc.nextInt();
String[] s1 = new String[n1];
for (int i = 0; i < n1; i++) {
s1[i] = sc.next();
}
int n2 = sc.nextInt();
String[] s2 = new String[n2];
for (int i = 0; i < n2; i++) {
s2[i] = sc.next();
}
System.out.println(User.countCommonStrings(s1, s2, n1, n2));

}
}
```

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

```
publicclass User {
publicstaticint countCommonStrings(String[] a, String[] b, int n1, int n2) {
int count = 0;
ArrayList<String> al = new ArrayList<String>();
for (int i = 0; i < n1; i++) {
for (int j = 0; j < n2; j++) {
if (a[i].equalsIgnoreCase(b[j]))
if (!al.contains(b[i])) {
count++;
al.add(a[i]);
}
}
}
return count;
}
}
```

```
import java.util.HashSet;
import java.util.Iterator;
public class Palindrome {
            public static int removeDuplicate(String[] words1,String[]
words2)
            {

                              int count=0;
                            HashSet<String> s1=new HashSet<String>();
                              HashSet<String> s2=new HashSet<String>();
                              for(int i=0;i<words1.length;i++)
                              {
                                      s1.add(words1[i]);

                              }
                              for(int i=0;i<words2.length;i++)
                              {
                                      s2.add(words2[i]);
                              }
                              Iterator<String> it1=s1.iterator();

                  while(it1.hasNext())
                              {
                                      String its1=it1.next();
                                      Iterator<String> it2=s2.iterator();
                                      while(it2.hasNext())
                                      {
                                              String its2=it2.next();
                                              if(its1.equals(its2))
                                              {
                                                     count++;
                                              }
                                              }
```

```
                        }
                return count;
                }
}
```

35. Sequence Sum

Write a program to read a non-negative integer n, and find sum of fibonanci series for n number..

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

The fibonacci seqence is a famous bit of mathematics, and it happens to have a recursive definition.

The first two values in the sequnce are 0 and 1.

Each subsequent value is the sum of the previous two values, so the whole seqence is 0,1,1,2,3,5 and so on.

You will have to find the sum of the numbers of the Fibonaaci series for a given int n.

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:

5

Sample Output 1:

Solution:

```
importjava.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
System.out.println(User.getFibonacciSum(n));
}
}


publicclass User {
publicstaticint getFibonacciSum(int n) {
int a = 0, b = 1, c = 0, d = 1;
for (int i = 3; i <= n; i++) {
c = a + b;
a = b;
b = c;
d = d + c;
}
return d;
}
}
```

36. Email Validation

Write a program to read a string and validate the given email-id as input.
Validation Rules:
1. Ensure that there are atleast 5 characters between '@' and '.'
2. There should be only one '.' and one '@' symbol.
3. The '.' should be after the '@' symbol.
4. There must be atleast three characters before '@'.
5. The string after '.' should only be 'com'

Include a class UserMainCode with a static method **ValidateEmail** which accepts the string. The return type is TRUE / FALSE as per problem.
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:
test@gmail.com
Sample Output 1:
TRUE

Sample Input 2:
academy@xyz.com
Sample Output 2:
FALSE

Solution:

```
import java.util.Scanner;

class Main {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
String email = sc.next();
System.out.println(User.ValidateEmail(email));
}
}


public class User {
public static boolean ValidateEmail(String email) {
boolean b = false;
if (email.matches("[a-zA-Z0-9]{3,}(@)[a-zA-Z]{5,}(.)(com)"))
b = true;
return b;
}
}
```

37. Symmetric Difference

Write a program to read two integer array and calculate the symmetric difference of the two arrays. Finally Sort the array.
Symmetric difference is the difference of A Union B and A Intersection B ie. [(A U B) - (A ^ B)]
Union operation merges the two arrays and makes sure that common elements appear only once.

Intersection operation includes common elements from both the arrays.
Ex - A={12,24,7,36,14} and B={11,26,7,14}.
A U B ={ 7,11,12,14,24,26,36} and
A ^ B = {7,14}
Symmetric difference of A and B after sorting= [A U B] - [A ^ B] = {11,12,24,26,36}.
Include a class UserMainCode with a static method **getSymmetricDifference** which accepts the integer array. The return type is an integer array.
Create a Class Main which would be used to accept the two integer arrays and call the static method present in UserMainCode.

Input and Output Format:
Input consists of an integer n which is the number of elements followed by n integer values. The same sequnce is followed for the next array.
Output consists of sorted symmetric difference array.
Refer sample output for formatting specifications.

Sample Input 1:
5
11
5
14
26
3
3
5
3
1
Sample Output 1:
1
11
14
26

Solution:

```
public class Main {
public static void main(String[] args) throws ParseException {
Scanner sc=new Scanner(System.in);

int n1=sc.nextInt();
int[] a=new int[n1];
for(int i=0;i<n1;i++)
```

```
a[i]=sc.nextInt();
int n2=sc.nextInt();
int[] b= new int[n2];
for(int i=0;i<n2;i++)
b[i]=sc.nextInt();
int[] res=User.display(a,b,n1,n2);
for(int i=0;i<res.length;i++)
System.out.println(res[i]);



}
}


public class User {
public static int[] display(int a[],int b[],int n1,int n2)
{
TreeSet<Integer> ts1=new TreeSet<Integer>();
TreeSet<Integer> ts2=new TreeSet<Integer>();
TreeSet<Integer> ts3=new TreeSet<Integer>();
ArrayList<Integer> aa=new ArrayList<Integer>();
for(int i=0;i<a.length;i++)
ts1.add(a[i]);
for(int i=0;i<b.length;i++)
ts2.add(b[i]);
ts1.addAll(ts2);
for(int i=0;i<n1;i++)
{
for(int j=0;j<n2;j++)
{
if(a[i]==b[j])
ts3.add(a[i]);
}
}
ts1.removeAll(ts3);
aa.addAll(ts1);
int res[]=new int[aa.size()];
for(int i=0;i<res.length;i++)
res[i]=aa.get(i);
return res;
}
}
```

38. Day of Week

Write a program to read a string containing date in DD/MM/YYYY format and prints the day of the week that date falls on.
Return the day in lowercase letter (Ex: monday)

Include a class UserMainCode with a static method **getDayOfWeek** which accepts the string. The return type is the 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:
02/04/1985
Sample Output 1:
tuesday

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

public class Main {
	public static void main(String[] args) throws ParseException {
		Scanner s=new Scanner(System.in);
		String s1=s.next();
		System.out.println(User.findOldDate(s1));
	}

}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class User {
	public static String findOldDate(String s1) throws ParseException
	{
		SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
		Date d1=sd1.parse(s1);
		SimpleDateFormat sd2=new SimpleDateFormat("EEEEE");
		String name=sd2.format(d1);
		return name.toLowerCase();
```

```
        }
}
```

39. Addtime

Write a program to read two String variables containing time intervals in hh:mm:ss format. Add the two time intervals and return a string in days:hours:minutes:seconds format where DD is number of days.
Hint: Maximum value for hh:mm:ss is 23:59:59

Include a class UserMainCode with a static method **addTime** which accepts the string values. The return type is the string.
Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
12:45:30
13:50:45
Sample Output 1:
1:2:36:15

Sample Input 2:
23:59:59
23:59:59
Sample Output 2:
1:23:59:58

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = newScanner(System.in);
```

```
String s1 = sc.next();
String s2 = sc.next();
System.out.println(User.addTime(s1, s2));
}
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;

publicclass User {
publicstatic String addTime(String s1, String s2) throws ParseException {
// adding two times
SimpleDateFormat sdf = newSimpleDateFormat("HH:mm:ss");
sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
Date d1 = sdf.parse(s1);
Date d2 = sdf.parse(s2);
long time = d1.getTime() + d2.getTime();
String s = sdf.format(new Date(time));
// to calculate the day
Calendar c = Calendar.getInstance();
c.setTime(sdf.parse(s));
int day = c.get(Calendar.DAY_OF_MONTH);
if (day > 1)
day = day - 1;
String op = day + ":" + s;
return op;
}
}
```

```
import java.util.StringTokenizer;


public class Palindrome {
            public static String removeDuplicate(String a,String b)
            {

                  StringTokenizer st1=new StringTokenizer(a,":");
                  StringTokenizer st2=new StringTokenizer(b,":");
```

```
                int h1=Integer.parseInt(st1.nextToken());
                int m1=Integer.parseInt(st1.nextToken());
                int s1=Integer.parseInt(st1.nextToken());
                int d=0;
                int h2=Integer.parseInt(st2.nextToken());
                int m2=Integer.parseInt(st2.nextToken());
                int s2=Integer.parseInt(st2.nextToken());

                int m,h,s;
                m=m1+m2;
                h=h1+h2;
                s=s1+s2;

                if(s>=60)
                {
                      m=m+1;
                      s=s-60;
                      if(m1>=60)
                      {
                            h=h+1;
                            m=m-60;
                            if(h>=24)
                            {
                                  d=d+1;
                                  h=h-24;
                            }
                      }
                }

                if(m1>=60)
                {
                      h=h+1;
                      m=m-60;
                      if(h>=24)
                      {
                            d=d+1;
                            h=h-24;
                      }
                }

                if(h>=24)
                {
                      d=d+1;
                      h=h-24;
                }

                StringBuffer sb=new StringBuffer();

      sb.append(d).append(":").append(h).append(":").append(m).append(":").ap
pend(s);

                return sb.toString();


                }
}
```

```
0:00:01
0:00:02
0:0:0:3


12:45:30
13:50:45
1:2:36:15
```

12:20:20

22:20:10

```
12:20:20
22:20:10
1:10:40:30
```

1:20:20

2:20:10

```
1:20:20
2:20:10
0:3:40:30
```

```
import java.util.StringTokenizer;

public class Palindrome {
            public static String removeDuplicate(String a,String b)
            {
                  StringTokenizer st1=new StringTokenizer(a,":");
                  StringTokenizer st2=new StringTokenizer(b,":");

                  int h1=Integer.parseInt(st1.nextToken());
                  int m1=Integer.parseInt(st1.nextToken());
                  int s1=Integer.parseInt(st1.nextToken());
                  int d=0;
                  int h2=Integer.parseInt(st2.nextToken());
                  int m2=Integer.parseInt(st2.nextToken());
                  int s2=Integer.parseInt(st2.nextToken());

                  int m,h,s;
                  m=m1+m2;
                  h=h1+h2;
                  s=s1+s2;

                  while(s>=60)
```

```
                {
                        m=m+1;
                        s=s-60;
                }
                while(m>=60)
                        {
                                h=h+1;
                                m=m-60;
                        }

                while(h>=24)
                        {
                                d=d+1;
                                h=h-24;
                        }

                StringBuffer sb=new StringBuffer();

        sb.append(d).append(":").append(h).append(":").append(m).append(":").ap
pend(s);

                return sb.toString();


                }
}
```

40. ISBN Validation

Write a program to read a string and validate the given ISBN as input.
Validation Rules:
1. An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book.
2. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third ..all the way until you add 1 times the last digit.
If the final number leaves no remainder when divided by 11 the code is a valid ISBN.
Example 1:
Input:0201103311
Calculation: 10*0 + 9*2 + 8*0 + 7*1 + 6*1 + 5*0 + 4*3 + 3*3 + 2*1 + 1*1= 55.
55 mod 11 = 0
Hence the input is a valid ISBN number
Output: true
Include a class UserMainCode with a static method **validateISBN** which accepts the string. The return type is TRUE / FALSE as per problem.

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:
0201103311
Sample Output 1:
TRUE

Solution:

```
import java.util.Scanner;

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

}
```

```
import java.util.*;

public class User {
	public static boolean validateISBN(String isbn)
	{
		int sum=0,k=10;
		for(int i=0;i<isbn.length();i++)
		{
			int a=Character.getNumericValue(isbn.charAt(i));
			sum=sum+(a*k);
			k--;
		}
		if(sum%11==0)
			return true;
		else
			return false;
	}
}
```

41. Date Format

Write a program to read two String variables in DD-MM-YYYY.Compare the two dates and return the older date in 'MM/DD/YYYY' format.

Include a class UserMainCode with a static method **findOldDate** which accepts the string values. The return type is the string.
Create a Class Main which would be used to accept the two string values and call the static method present in UserMainCode.

Input and Output Format:
Input consists of two string.
Output consists of a string.
Refer sample output for formatting specifications.

Sample Input 1:
05-12-1987
8-11-2010
Sample Output 1:
12/05/1987

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

public class Main {
	public static void main(String[] args) throws ParseException {
		Scanner s=new Scanner(System.in);
		String s1=s.next();
		String s2=s.next();
		System.out.println(User.findOldDate(s1,s2));
	}

}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;

public class User {
	public static String findOldDate(String s1,String s2) throws ParseException
	{
		SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
```

```
		Date d1=sd1.parse(s1);
		Date d2=sd1.parse(s2);
		Calendar c=Calendar.getInstance();
		c.setTime(d1);
		int day1=c.get(Calendar.DAY_OF_MONTH);
		int m1=c.get(Calendar.MONTH);
		int y1=c.get(Calendar.YEAR);
		c.setTime(d2);
		int day2=c.get(Calendar.DAY_OF_MONTH);
		int m2=c.get(Calendar.MONTH);
		int y2=c.get(Calendar.YEAR);
		SimpleDateFormat sd2=new SimpleDateFormat("MM/dd/yyyy");
		String res=null;
		if(y1==y2)
		{
			if(m1==m2)
			{
				if(day1==day2)
				{
					res=sd2.format(d1);
				}
			}
				else
				{
					if(m1>m2)
						res=sd2.format(d2);
					else
						res=sd2.format(d1);
				}
			}
		else
		{
			if(y1>y2)
				res=sd2.format(d2);
			else
				res=sd2.format(d1);
		}

		return res;
	}
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;

import java.util.Date;
public class Palindrome {
		public static String removeDuplicate(String s1,String s2) throws
ParseException
```

```
{
        SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
        Date d1=sd1.parse(s1);
        Date d2=sd1.parse(s2);
        String res=null;

        SimpleDateFormat sfd2=new SimpleDateFormat("MM/dd/yyyy");


        if(d1.compareTo(d2)<0)
        {
              res=sfd2.format(d1);
        }
        else
        {
              res=sfd2.format(d2);
        }

        return res;

        }
}
```

42. Interest calculation

1. Read account details from the User. The details would include id, DOB (date of birth) and amount in the given order. The datatype for id is string, DOB is string and amount is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and amount as value.
3. Rate of interest as on 01/01/2015:
   a. If the age greater than or equal to 60 then interest rate is 10% of Amount.
   b.If the age less then to 60 and greater than or equal to 30 then interest rate is 7% of Amount.
   v. If the age less then to 30 interest rate is 4% of Amount.
4. Revised Amount= principle Amount + interest rate.
5. You decide to write a function **calculateInterestRate** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:
Input consists of account details. The first number indicates the size of the acoount. The next three values indicate the user id, DOB and amount. The Employee DOB format is “dd-mm-yyyy”

Output consists of the user id and the amount for each user one in a line.
Refer sample output for formatting specifications.

Sample Input 1:
4
SBI-1010
20-01-1987
10000
SBI-1011
03-08-1980
15000
SBI-1012
05-11-1975
20000
SBI-1013
02-12-1950
30000
Sample Output 1:
SBI-1010:10400
SBI-1011:16050
SBI-1012:21400
SBI-1013:33000

43. Discount rate calculation

Write a program to calculate discount of the acccount holders based on the transaction amount and registration date using below mentioned prototype:
1. Read account details from the User. The details would include id, DOR (date of registration) and transaction amount in the given order. The datatype for id is string, DOR is string and transaction amount is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOR as value, and the second hashmap contains same employee ids as key and amount as value.
3. Discount Amount as on 01/01/2015:
   a. If the transaction amount greater than or equal to 20000 and registration greater than or equal to 5 year then discount rate is 20% of transaction amount.
   b. If the transaction amount greater than or equal to 20000 and registration less then to 5 year then discount rate is 10% of transaction amount.
   c. If the transaction amount less than to 20000 and registration greater than or equal to 5 year then discount rate is 15% of transaction amount.
   d. If the transaction amount less than to 20000 and registration less then to 5 year then discount rate is 5% of transaction amount.
4. You decide to write a function **calculateDiscount** which takes the above hashmaps as input

and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of transaction details. The first number indicates the size of the employees. The next three values indicate the user id, user DOR and transaction amount. The DOR (Date of Registration) format is “dd-mm-yyyy”

Output consists of a string which has the user id and discount amount one in a line for each user. Refer sample output for formatting specifications.

Sample Input 1:

4
A-1010
20-11-2007
25000
B-1011
04-12-2010
30000
C-1012
11-11-2005
15000
D-1013
02-12-2012
10000

Sample Output 1:

A-1010:5000
B-1011:3000
C-1012:2250
D-1013:500

Solution:

```
public class main {
public static void main(String []args){
Scanner sc=new Scanner(System.in);
int s=Integer.parseInt(sc.nextLine());
HashMap<String,String>hm=new HashMap<String,String>();
HashMap<String,Integer>hm1=new HashMap<String,Integer>();
for(int i=0;i<s;i++)
{
String id=sc.nextLine();
hm.put(id, sc.nextLine());
```

```
hm1.put(id,Integer.parseInt(sc.nextLine()));
}
TreeMap<String,Integer>tm=new TreeMap<String,Integer>();
tm=Usermaincode.findDiscountRate(hm,hm1);
Iterator<String> it=tm.keySet().iterator();
while(it.hasNext())
{
String n=it.next();
int fac=tm.get(n);
System.out.println(n+":"+fac);
}
}



public class UserMaincode
{
public static TreeMap<String,Integer> findDiscountRate
(HashMap<String,String>hm,HashMap<String,Integer>hm1) throws ParseException
{
TreeMap<String,Integer> tm=new TreeMap<String,Integer>();
SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
Iterator<String> itr1=hm.keySet().iterator();
while(itr1.hasNext())
{
        try
        {
        String id=itr1.next();
        String dor=hm.get(id);
        int am=hm1.get(id);
        Date d1=sdf.parse(dor);
        String s1="01-01-2015";
        Date d2=sdf.parse(s1);
        int y1=d1.getYear();
        int m1=d1.getMonth();
        int day1=d1.getDay();
        int y2=d2.getYear();
        int m2=d2.getMonth();
        int day2=d2.getDay();
        int exp=Math.abs(y1-y2);
        if(m1==m2)
        {
                if(day2>day1)
```

```
                        exp--;
                }
        if(m2>m1)
                exp--;
        if(am>=20000 && exp>=5)
        {
                int dis=(int) (0.20*am);
                tm.put(id,dis);
        }
        else if(am>=20000 && exp<5)
        {
                int dis=(int) (0.1*am);
                tm.put(id,dis);
        }
        else if(am<20000 && exp>=5)
        {
                int dis=(int) (0.15*am);
                tm.put(id,dis);
        }
        else if(am<20000 && exp<5)
        {
                int dis=(int) (0.05*am);
                tm.put(id,dis);
        }
        }
        catch(Exception e){
                System.out.println(e);
        }
}
return tm;
}
}
}
}
```