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.

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

**int** r=UserMainCode.c*heckSum*(n);

**if**(r==1)

              {

                     System.***out***.println("The sum of odd digits are odd");

              }

**else**

              {

                     System.***out***.println("The sum of odd digits are even");

              }

              s.close();

       }

}

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** **int** checkSum(**int** n)

       {

**int** n1;

**int** sum=0;

**int** r;

**while**(n!=0)

              {

                     n1=n%10;

**if**(n1%2!=0)

                     {

                           sum=sum+n1;

                     }

                     n=n/10;

              }

**if**(sum%2==0)

              {

                     r=-1;

              }

**else**

              {

                     r=1;

              }

**return** r;

       }

}

**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

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

              String number=s.next();

**int** r=UserMainCode.*validateNumber*(number);

**if**(r==1)

              {

                     System.***out***.println("Valid number format“);

              }

**else**

              {

                     System.***out***.println("Invalid number format");

              }

              s.close();

       }

}

**USERMAINCODE:**

**import** java.util.\*;

**public** **class** UserMainCode {

**public** **static** **int** validateNumber(String number)

       {

**int** b;

**if**(number.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

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

              System.***out***.println(UserMainCode.*sumOfSquaresOfEvenDigits*(n));

              s.close();

       }

}

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** **int** sumOfSquaresOfEvenDigits(**int** n)

       {

**int** n1=0;

**int** sum=0;

**while**(n!=0)

              {

                     n1=n%10;

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

                     {

                           sum+=n1\*n1;

                     }

                     n=n/10;

              }

**return** sum;

       }

}

**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

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

              String str=s.nextLine();

              System.***out***.println(UserMainCode.*getMiddleChars*(str));

             s.close();

       }

}

**USERMAINCODE:**

**import** java.util.\*;

**public** **class** UserMainCode {

**public** **static** String getMiddleChars(String str)

       {

              StringBuffer sb=**new** StringBuffer();

**if**(str.length()%2==0)

              {

              sb.append(str.substring((str.length()/2)-1,(str.length()/2)+1));

              }

**return** 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

**MAIN:**

**import** java.util.\*;

**public** **class** main {

**public** **static** **void** main(String[] args)

{

Scanner s=**new** Scanner(System.***in***);

String input=s.nextLine();

**int** r=UserMainCode.*checkCharacters*(input);

**if**(r==1)

{

System.***out***.println("Valid");

}

**else**

{

System.***out***.println("Invalid");

}

s.close();

}

}

**USERMAINCODE:**

**import** java.util.\*;

**public** **class** UserMainCode {

**public** **static** **int** checkCharacters(String input)

{

**int** r;

StringTokenizer t = **new** StringTokenizer(input," ");

String s = t.nextToken();

String s1 =s ;

**while**(t.hasMoreTokens())

{

s1 = t.nextToken();

}

**if**(s.charAt(0) == s1.charAt(s1.length()-1))

r=1;

**else**

r=0;

**return** r;

}

}

**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

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

              String s1=s.nextLine();

**int** n1=s.nextInt();

              System.***out***.println(UserMainCode.*formNewWord*(s1,n1));

             s.close();

       }

}

**USERMAINCODE:**

**import** java.util.\*;

**public** **class** UserMainCode {

**public** **static** String formNewWord(String s1,**int** n)

       {

              String s = **new** String();

**if**(s1.length()>n)

              {

              s = s1.substring(0,n) + s1.substring(s1.length()-n, s1.length());

**return** s;

              }

**else**

**return** **null**;

       }

}

**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

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

              System.***out***.println(UserMainCode.*reverseNumber*(n));

              s.close();

       }

}

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** **int** reverseNumber(**int** n)

       {

**int** a,r=0;

**while**(n!=0)

              {

                     a=n%10;

                     r=r\*10+a;

                     n=n/10;

              }

**return** r;

       }

}

**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**

**Main:**

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

              ArrayList<Integer> list1=**new** ArrayList<Integer>();

              ArrayList<Integer> list2=**new** ArrayList<Integer>();

              ArrayList<Integer> newlist=**new** ArrayList<Integer>();

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

              {

                     list1.add(s.nextInt());

              }

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

              {

                     list2.add(s.nextInt());

              }

              newlist=UserMainCode.*sortMergedArraylist*(list1,list2);

**for** (**int** i = 0; i < 3; i++)

              {

                     System.***out***.println(newlist.get(i));

              }

              s.close();

       }

}

**UERMAINCODE:**

**import** java.util.\*;

**public** **class** UserMainCode {

**public** **static** ArrayList<Integer> sortMergedArraylist(ArrayList<Integer> list1,ArrayList<Integer> list2)

       {

              list1.addAll(list2);

              Collections.*sort*(list1);

              ArrayList<Integer> ans=**new** ArrayList<Integer>();

              ans.add(list1.get(2));

              ans.add(list1.get(6));

              ans.add(list1.get(8));

**return** ans;

       }

}

**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

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

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

String s1=sc.nextLine();

**int** b=UserMainCode.*ValidateDate*(s1);

**if**(b==1){

       System.***out***.println("Valid date format");

}

**else**{

       System.***out***.println("Invalid date format");

}

sc.close();

}}

**UserMainCode:**

**import java.util.\*;**

**import java.text.\*;**

**public class UserMainCode{**

**public static int ValidateDate(String s1) {**

**if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))**

**{**

**SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");**

**sdf.setLenient(false);**

**try {**

**Date d1=sdf.parse(s1);**

**return 1;**

**} catch (ParseException e) {**

**return -1;**

**}**

**}**

**else{**

**return -1;}}}**

**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

**Main:**

**import** java.util.\*;

**public** **class** Main{

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

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

String str=sc.nextLine();

**int** b=UserMainCode.*ValidateTime*(str);

**if**(b==1){

       System.***out***.println("Valid time");

}

**else**{

       System.***out***.println("Invalid time");

}

sc.close();

}}

**UserMainCode:**

**import java.text.\*;**

**import java.util.\*;**

**public class UserMainCode{**

**public static int ValidateTime(String str){**

**StringTokenizer st=new StringTokenizer(str,":");**

**if(st.countTokens()==3)**

**{**

**SimpleDateFormat sdf1 = new SimpleDateFormat("h:mm:ss a");**

**sdf1.setLenient(false);**

**try**

**{**

**Date d2=sdf1.parse(str);**

**return 1;**

**}**

**catch(Exception e)**

**{**

**return -1;**

**}}**

**else**

**{**

**SimpleDateFormat sdf = new SimpleDateFormat("h:mm a");**

**sdf.setLenient(false);**

**try**

**{**

**Date d1=sdf.parse(str);**

**return 1;**

**}**

**catch(Exception e){**

**return -1;**

**}}}}**

**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

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

    String s1=s.next();

    System.***out***.println(UserMainCode.*encrypt*(s1));

    s.close();

}

}

**UserMainCode:**

**public** **class** UserMainCode{

**public** **static** String encrypt(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)

**if**((c==122)&&(i==0)){

c='A';}

**else**

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

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

    String password=s.next();

**int** b=UserMainCode.*ValidatePassword*(password);

**if**(b==1){

       System.***out***.println("Valid Password");

    }

**else**{

       System.***out***.println("Invalid Password");

    }

    s.close();

}}

**UserMainCode:**

**public** **class** UserMainCode{

**public** **static** **int** ValidatePassword(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

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

System.***out***.println(UserMainCode.*removeEvenVowels*(s1));

s.close();

}}

**UserMainCode:**

**public** **class** UserMainCode{

**public** **static** String removeEvenVowels(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

**Main:**

**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.*getSumOfPower*(n,a));

sc.close();

}}

**UserMainCode:**

**public** **class** UserMainCode{

**public** **static** **int** getSumOfPower(**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

**Main:**

**import** java.util.\*;

**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.*getBigDiff*(a,n));

              sc.close();

              }}

**UserMainCode:**

**import** java.util.\*;

**public** **class** UserMainCode{

**public** **static** **int** getBigDiff(**int** [] a,**int** n)

{

       Arrays.*sort*(a);

**int** n1=a[a.length-1]-a[0];

**return** n1;

}}

**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

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

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

**int** fr=sc.nextInt();

String a[]=**new** String[fr];

**for**(**int** i=0;i<fr;i++)

{

a[i]=sc.next();

}

String ba=sc.next();

UserMainCode.getElementPosition(a,ba);

sc.close();

}}

**UserMainCode:**

**import** java.util.\*;

**public** **class** UserMainCode{

**public** **static** **void** getElementPosition(String[] a, String b) {

ArrayList<String>al=**new** ArrayList<String>();

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

{

al.add(a[i]);

}

Collections.*sort*(al);

Collections.*reverse*(al);

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

{

**if**(b.equals(al.get(i)))

{

System.***out***.println(i+1);

}}}}

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

       System.***out***.println(UserMainCode.*addSeries*(n));

       s.close();

       }

}

**UserMainCode**

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** UserMainCode {

**public** **static** **int** addSeries(**int** n){

       List<Integer> l1=**new** ArrayList<Integer>();

**for**(**int** i=1;i<=n;i++)

**if**(i%2!=0)

       l1.add(i);

**int** n1=l1.get(0);

**for**(**int** i=1;i<l1.size();i++)

**if**(i%2!=0)

       n1=n1+l1.get(i);

**else**

       n1=n1-l1.get(i);

**return** n1;

       }

}

**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

**Main**

**import** java.util.Scanner;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

       String input1=s.next();

       String input2=s.next();

**int** input3=s.nextInt();

       System.***out***.println(UserMainCode.*calculateElectricityBill*(input1,input2,input3));

       s.close();

       }

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** calculateElectricityBill(String input1, String input2, **int** input3)

       {

**int** n1=Integer.*parseInt*(input1.substring(5, input1.length()));

**int** n2=Integer.*parseInt*(input2.substring(5, input2.length()));

**int** n=Math.*abs*((n2-n1)\*input3);

**return** n;

       }

       }

**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

**Main**

**import** java.util.Scanner;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.next();

UserMainCode.*sumOfDigits*(s1);

s.close();

}

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **void** sumOfDigits(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

**Main**

**import** java.util.Scanner;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.next();

String s2=s.next();

UserMainCode.*concatstring*(s1,s2);

s.close();

}

} **UserMainCode**

**public** **class** UserMainCode {

**public** **static** **void** concatstring(String s1, String s2) {

              StringBuffer sb=**new** StringBuffer();

**int** l1=s1.length();

**int** l2=s2.length();

**if**(l1==l2)

              {

              sb.append(s1).append(s2);

              }

**else** **if**(l1>l2)

              {

              sb.append(s1.substring(s1.length()-s2.length(),s1.length())).append(s2);

              }

**else** **if**(l1<l2)

              {

              sb.append(s1).append(s2.substring(s2.length()-s1.length(),s2.length()));

              }

              System.***out***.println(sb);

              }

}

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

       String s1=s.next();

**int** b=UserMainCode.*validateColorCode*(s1);

**if**(b==1)

       System.***out***.println("Valid");

**else**

       System.***out***.println("Invalid");

       s.close();

       }

       }

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** validateColorCode(String s1) {

**int** b=0,b1=0;

              String s2=s1.substring(1,s1.length());

**if**(s1.length()==7)

**if**(s1.charAt(0)=='#')

              b1=1;

**if**(b1==1){

              /\*for(int i=0;i<s2.length();i++){

              char c=s2.charAt(i);

              if(c!='#')

              {\*/

**if**(s2.matches("[A-F0-9]{1,}"))

              b=1;

**else**

              b=-1;

              //break;

              }

**return** b;

              }

       }

**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

**Main**

**import** java.util.Scanner;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.next();

**int** b=UserMainCode.*validatestrings*(s1);

**if**(b==1){

       System.***out***.println("Valid");}

**else**

       System.***out***.println("Invalid");

       s.close();

}

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** validatestrings(String s1) {

**int** res=0;

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

              {

              res=1;

              }

**else**

              res=-1;

**return** res;

              }

              }

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

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

**int** s=sc.nextInt();

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

**for**(**int** i=0;i<s;i++){

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

}

System.***out***.println(UserMainCode.*sizeOfResultandHashMap*(hm));

sc.close();

} }

**UserMainCode**

**import** java.util.HashMap;

**import** java.util.Iterator;

**public** **class** UserMainCode {

**public** **static** **int** sizeOfResultandHashMap(HashMap<Integer, String> hm) {

**int** count=0;

              Iterator<Integer>itr=hm.keySet().iterator();

**while**(itr.hasNext())

              {

**int** n=itr.next();

**if**(n%4!=0)

              {

              count++;

              }

              }

**return** count;

              }

              }

**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

**Main**

**import** java.util.\*;

**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();

              }

              System.***out***.println(UserMainCode.*checkLargestAmongCorner*(a));

              s.close();

       }

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** checkLargestAmongCorner(**int** []a)

       {

**int** max=0;

**int** x,y,z;

       x=a[0];

       y=a[a.length/2];

       z=a[a.length-1];

**if**(x>y && x>z)

       max=x;

**else** **if**(y>x && y>z)

       max=y;

**else** **if**(z>x && z>y)

       max=z;

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

                     Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

**int** r=s.nextInt();

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

              }

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** calculateNcr(**int** n, **int** r) {

**int** fact=1,fact1=1,fact2=1;

**for**(**int** i=1;i<=n;i++)

              {

              fact=fact\*i;

              }

              //System.out.println(fact);

**for**(**int** i=1;i<=r;i++)

              {

              fact1=fact1\*i;

              }

              //System.out.println(fact1);

**for**(**int** i=1;i<=(n-r);i++)

              {

              fact2=fact2\*i;

              }

              //System.out.println(fact2);

**int** res=fact/(fact1\*fact2);

**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

**Main**

**import** java.util.Scanner;

**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.*getSumOfIntersection* (a,b,n,m);

**if**(u==-1)

       System.***out***.println("No common elements");

**else**

       System.***out***.println(u);

sc.close();

}}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** getSumOfIntersection(**int** a[],**int** b[],**int** n,**int** m)

              {

**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

**Main**

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main(String[] args)

{

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

String a=sc.next();

**int** e=UserMainCode.*validatePassword*(a);

**if**(e==1){

System.***out***.println("Valid");

}

**else**

{

       System.***out***.println("Invalid");

}

sc.close();

}}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** validatePassword(String a){

**int** d=0;

**if**(a.length()>=8){

**if**(a.contains("#") || a.contains("@") || a.contains("\_"))

       {

**char** c= a.charAt(0);

       //System.out.println(c);

**if**(Character.*isAlphabetic*(c))

       {

**char** dd=a.charAt(a.length()-1);

       //System.out.println(dd);

**if**((Character.*isAlphabetic*(dd))||(Character.*isDigit*(dd)))

       {

**if**(a.matches(".\*[0-9]{1,}.\*")||a.matches(".\*[a-zA-Z]{1,}.\*")){

       d=1;

       }

       }

       }

       }

       }

**else**

       d=-1;

**return** d;

}}

**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

**Main**

**import** java.util.\*;

**public** **class** Main3 {

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

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

String s1=sc.next();

String s2=sc.next();

**boolean** b=UserMainCode3.*validateIDLocations*(s1,s2);

**if**(b==**true**)

System.***out***.println("Valid id");

**else**

System.***out***.println("Invalid id");

sc.close();

}

}

**UserMainCode**

**import** java.util.StringTokenizer;

**public** **class** UserMainCode3 {

**public** **static** **boolean** validateIDLocations(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

**Main**

**import** java.util.\*;

**public** **class** Main

{

**public** **static** **void** main(String[] args)

{

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

**int** n=Integer.*parseInt*(sc.nextLine());

String[] a=**new** String[n];

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

a[i]=sc.nextLine();

**int** m=Integer.*parseInt*(sc.nextLine());

System.***out***.println(UserMainCode.*removeElements*(a,m));

sc.close();

}

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** removeElements(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

**Main**

**import** java.text.\*;

**import** java.util.\*;

**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(UserMainCode.*getMonthDifference*(s1,s2));

sc.close();

}}

**UserMainCode**

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**public** **class** UserMainCode {

**public** **static** **int** getMonthDifference(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

**Main**

**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.*addEvenOdd*(a));

sc.close();

}

}

**UserMainCode**

**public** **class** UserMainCode6 {

**public** **static** **int** addEvenOdd(**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

**Main**

**import** java.util.\*;

**public** **class** Main7 {

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

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

       String ipAddress=sc.next();

**boolean** b=UserMainCode7.*validateIpAddress*(ipAddress);

**if**(b==**true**)

       System.***out***.println("Valid");

**else**

       System.***out***.println("Invalid");

       sc.close();

       }

       }

**UserMainCode**

**import** java.util.StringTokenizer;

**public** **class** UserMainCode7 {

**public** **static** **boolean** validateIpAddress(String ipAddress) {

**boolean** b1=**false**;

              StringTokenizer t=**new** StringTokenizer(ipAddress,".");

**int** a=Integer.*parseInt*(t.nextToken());

**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

**Main**

**import** java.text.ParseException;

**import** java.util.\*;

**public class** Main {

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

{

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

String s2=s.nextLine();

**int** output=UserMainCode.*getDateDifference*(s1,s2);

System.***out***.println(output);

s.close();

}

}

**UserMainCode**

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

**public class** UserMainCode {

**public static int** getDateDifference(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;

}

}

**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

**Main**

**import** java.util.\*;

**public class** Main {

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

Scanner s=**new** Scanner(System.***in***);

System.***out***.println("enter the string");

String s1=s.nextLine();

String output=UserMainCode.*fileIdentifier*(s1);

System.***out***.println(output);

s.close();

}

}

**UserMainCode**

**import** java.util.\*;

**public class** UserMainCode {

**public static** String fileIdentifier(String s1)

{

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

t.nextToken();

String s2=t.nextToken();

**return** s2;

}

}

**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

**Main**

**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();

StringBuffer sb1=**new** StringBuffer(s1.replace(" ",""));

StringBuffer sb2=**new** StringBuffer(s2.replace(" ",""));

**int** output=UserMainCode.*commonChars*(s1,s2,sb1,sb2);

System.***out***.println(output);

sc.close();

}

}

**UserMainCode**

**import** java.util.\*;

**public class** UserMainCode {

**public static int** commonChars(String s1,String s2,StringBuffer sb1,StringBuffer sb2) {

**for**(**int** i=0;i<sb1.length();i++){

**int** c=0;

**for**(**int** j=i+1;j<sb1.length();j++){

**if**(sb1.charAt(i)==sb1.charAt(j)){

sb1.deleteCharAt(j);

c++;

}

}

**if**(c>=1){

sb1.deleteCharAt(i);

}

}

**for**(**int** i=0;i<sb2.length();i++){

**int** c=0;

**for**(**int** j=i+1;j<sb2.length();j++){

**if**(sb2.charAt(i)==sb2.charAt(j)){

sb2.deleteCharAt(j);

c++;

}

}

**if**(c>=1){

sb2.deleteCharAt(i);

}

}

**int** count=0;

**for**(**int** i=0;i<sb1.length();i++){

**for**(**int** j=0;j<sb2.length();j++){

**if**(sb1.charAt(i)==sb2.charAt(j)){

count++;

}

}

}

**return** (count);

}

}

**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

**Main**

**import** java.util.\*;

**public class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

String output=UserMainCode.*nameFormatter*(s1);

System.***out***.println(output);

s.close();

}

}

**UserMainCode**

**import** java.util.\*;

**public class** UserMainCode {

**public static** String nameFormatter(String s1) {

StringBuffer sb=**new** StringBuffer();

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

String s2=st.nextToken();

String s3=st.nextToken();

sb.append(s3).append(",");

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

**return** sb.toString();

}

}

**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

**Main**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

String s1=s.nextLine();

String c=s.nextLine();

String output=UserMainCode.removeCharacter(s1,c);

System.out.println(output);

}

}

**UserMainCode**

**import** java.util.\*;

**public class** UserMainCode {

**public static** String removeCharacter(String s1,String c)

{

String d=s1.replace(c,"");

**return** d;

}

}

**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

**Main**

**import** java.util.\*;

**public class** User {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

String s2=s1.toLowerCase();

**int** output=UserMainCode.*getVowels*(s2);

//System.out.println(output);

**if**(output==1)

{

System.***out***.println("yes");

}

**else**

System.***out***.println("no");

s.close();

}

}

**UserMainCode**

**import** java.util.\*;

**public class** UserMainCode {

**public static int** getVowels(String s2) {

**if**(s2.contains("a") && s2.contains("e") && s2.contains("i") && s2.contains("o") && s2.contains("u") )

{

**return** 1;

}

**else**

**return** -1;

}

}

**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

**Main**

**import** java.util.\*;

**public class** Main

{

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

String output=UserMainCode.*swapCharacter*(s1);

System.***out***.println(output);

s.close();

}

}

**UserMainCode**

**import** java.util.\*;

**public class** UserMainCode {

**public static** String swapCharacter(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).append(a);

}

**return** sb.toString();

}

**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));

**return** sb.toString();

}

}

}

**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

**Main**

**import** java.util.HashMap;

**import** java.util.Scanner;

**public class** Main {

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

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

**int** s=sc.nextInt();

HashMap<Integer,Float>hm=**new** HashMap<Integer,Float>();

**for**(**int** i=0;i<s;i++)

{

**int** r=sc.nextInt();

Float j=sc.nextFloat();

hm.put(r,j);

}

System.***out***.println(UserMainCode.*display*(hm));

sc.close();

}

}

**UserMainCode**

**import** java.text.DecimalFormat;

**import** java.util.\*;

**public class** UserMainCode

{

**public static** String display(HashMap<Integer,Float>hm)

{

**float** sum=0;

**int** count=0;

DecimalFormat df=**new** DecimalFormat("#.00");

Iterator<Integer> it=hm.keySet().iterator();

**while**(it.hasNext())

{

**int** y=it.next();

**if**(y%2==0)

{

sum=(**float**) (sum+hm.get(y));

count++;

}}

**float** d=sum/count;

**return** df.format(d);

}

}

**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

aHashMap<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

**Main**

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] arg)

{

HashMap<Integer,Double> hm=**new** HashMap<Integer,Double>();

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

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

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

{

**int** a=sc.nextInt();

**double** s=sc.nextDouble();

hm.put(a,s);

}

System.***out***.println(UserMaincode.*dis*(hm));}}

**UserMainCode**

**class** UserMaincode

{

**public** **static** **double** dis(HashMap<Integer,Double> h1)

{

**double** avg=0.0,sum=0.0;

**int** k=0;

**for**(Map.Entry m:h1.entrySet())

{

**int** a=(Integer)m.getKey();

**if**(a%2!=0)

{

Double d=(Double) m.getValue();

sum=sum+d;

k++;

}

}

avg = (**double**)sum/k;

DecimalFormat df = **new** DecimalFormat(".##");

String b1 = df.format(avg);

**double** b = Double.*parseDouble*(b1);

**return** b;}

**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

**Main**

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

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

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

       String input1=sc.next();

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

}

}

**UserMainCode**

**class** UserMainCode

{

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

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

String s1=sc.nextLine();

System.***out***.println(UserMainCode.*largestChunk*(s1));

}

}

**UserMaincode**

**class** UserMainCode

{

**public** **static** **int** largestChunk(String s1) {

**int** max=1;

**int** b=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;

b=max+1;

}

}

**return** b;

}

}

**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

**Main**

**import** java.util.\*;

**import** java.text.\*;

**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.*uniqueCounter*(s1));

}}

**UserMainCode**

**class** UserMaincode

{

**public** **static** **int** uniqueCounter(String s1)

       {

StringBuffer sb = **new** StringBuffer(s1);

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

**int** count = 0;

**for** (**int** j = i + 1; j < sb.length(); j++) {

**if** (sb.charAt(i) == sb.charAt(j)) {

sb.deleteCharAt(j);

j--;

count++;

}

}

**if** (count >= 1) {

sb.deleteCharAt(i);

i--;

}

}

**return** sb.length();

}

}

**45.Name Shrinking**

Write a program that accepts a string as input and converts the first two names into dotseparated

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

**Main**

**import** java.text.\*;

**import** java.util.\*;

**public** **class** Main {

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

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

       String s1=sc.nextLine();

System.***out***.println(UserMainCode.*getFormatedString*(s1));

}

}

**UserMainCode**

**class** UserMainCode

{

**public** **static** String getFormatedString(String s1) {

StringBuffer sb=**new** StringBuffer();

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

String s2=st.nextToken();

String s3=st.nextToken();

String s4=st.nextToken();

sb.append(s4).append(" ");

sb.append(s3.substring(0,1));

sb.append(".");

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

**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)**

**Main**

**import** java.util.Scanner;

**public** **class** Main {

**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(UserMainCode.oddDigitSum(s2));

}}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **int** oddDigitSum (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

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main{

**public** **static** **void** main(String[]args)

{**int** j=0;

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

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

j=UserMainCode.*getUnique*(n);

**if**(j>0)

{

System.***out***.println("Not Unique");

}

**else** **if**(j==0)

{

System.***out***.println("Unique");

}

}}

**class** UserMainCode

{

**public** **static** **int** getUnique(**int** n)

{

**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++;

}

}}

**return** count;

}

}

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

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

**int** n=Integer.*parseInt*(sc.nextLine());

HashMap<Integer,Integer>h1=**new** HashMap<Integer,Integer>();

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

{

h1.put(sc.nextInt(),sc.nextInt());

}

System.***out***.println(UserMainCode.*getLowest*(h1));

}

}

**UserMainCode**

**class** UserMainCode {

**public** **static** **int** getLowest(HashMap<Integer,Integer>h1)

{

ArrayList<Integer>a1=**new** ArrayList<Integer>();

**int** m=0;

Iterator<Integer> it=h1.keySet().iterator();

**while**(it.hasNext())

{

**int** x=it.next();

a1.add(h1.get(x));

}

Collections.*sort*(a1);

m=a1.get(0)+a1.get(1)+a1.get(2);

**return** m;

}}

**49.Color Code Validation same as 21**

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

**Main**

**import** java.util.\*;

**public** **class** Add {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.next();

**boolean** b=userMainCode.validateColourCode (s1);

**if**(b==**true**)

System.***out***.println("valid color code");

**else**

System.***out***.println("invalid color code");

}

**UserMainCode**

**static** **class** userMainCode{

**public** **static** **boolean** validateColourCode (String s1)

**boolean** b=**false**;

**if**(s1.length()==7&&s1.matches("#[A-F0-9]{1,}")){

b=**true**;

}

**return** b;

}}}

**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

**Main**

**import** java.util.\*;

**public** **class** Main {

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

Scanner s= **new** Scanner(System.***in***);

String input= s.next();

**int** n=s.nextInt();

System.***out***.println(userMainCode.*getString*(input,n));

}

}

**UserMainCode**

**class** userMainCode {

**public** **static** String getString(String input, **int** n){

StringBuffer sb=**new** StringBuffer();

sb.append(input);

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

sb.append(input.substring(input.length()-n,input.length()));

}

**return** sb.toString();

}

}

**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

**Main**

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

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

Scanner s= **new** Scanner(System.***in***);

String input= s.next();

System.***out***.println(userMainCode.*calculateBornDay*(input));

}

}

**UserMainCode**

**class** userMainCode{

**public** **static** String calculateBornDay(String input) **throws** ParseException{

SimpleDateFormat sdf= **new** SimpleDateFormat("dd-MM-yyyy");

SimpleDateFormat sdf1= **new** SimpleDateFormat("EEEEE");

Date d= sdf.parse(input);

String s1= sdf1.format(d);

**return** s1;

}

}

**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

**Main**

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

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

Scanner s= **new** Scanner(System.***in***);

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

**int** n= s.nextInt();

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

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

}

System.***out***.println(UserMainCode.*afterDelete*(hm));

s.close();

}

}

**UserMainCode**

**class** UserMainCode{

**public** **static** **int** afterDelete(HashMap<Integer, String> hm){

**int** count=0;

Iterator<Integer>itr=hm.keySet().iterator();

**while**(itr.hasNext())

{

**int** n=itr.next();

**if**(n%3!=0)

{

count++;

}

}

**return** count;

}

}

**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

**Main**

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

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

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

String a=sc.next();

String b=sc.next();

**int** c=sc.nextInt();

**long** res=(userMainCode.*calculateExperience*(a,b,c));

**if**(res==c)

{

System.***out***.println("true");

}

**else**

System.***out***.println("false");

}

}

**UserMainCode**

**class** userMainCode{

**public** **static** **long** calculateExperience(String a, String b, **int** c)**throws** ParseException{

SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

Date d=**new** Date();

Date d1=**new** Date();

d=sdf.parse(a);

d1=sdf.parse(b);

**long** t=d.getTime();

**long** t1=d1.getTime();

**long** t3=t1-t;

**long** l1=(24 \* 60 \* 60 \* 1000);

**long** l=l1\*365;

**long** res=t3/l;

**return** res;

}

}

**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

**Main**:

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

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

Scanner s= **new** Scanner(System.***in***);

String input=s.next();

System.***out***.println(UserMainCode.*getSpecialChar*(input));

}

}

**UserMainCode**:

**class** UserMainCode{

**public** **static** String getSpecialChar(String input){

**int** i;

StringBuffer sb= **new** StringBuffer();

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

{

**char** a=input.charAt(i);

**if** (!Character.*isAlphabetic*(a))

sb.append(a);

}

**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

class Main{

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

Scanner s= **new** Scanner(System.***in***);

System.out.println(“enter a string”);

String input= s.next();

**int** n= s.nextInt();

System.***out***.println(UserMainCode.*repeatString*(input, n));

}

}

**class** UserMainCode{

**public** **static** String repeatString(String input, **int** n){

StringBuffer sb= **new** StringBuffer();

String s1= **new** String();

**if** (n==1){

s1=input.substring(0,1);

sb.append(s1).append(" ");

}

**if**(n==2){

s1=input.substring(0,2);

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

sb.append(s1).append(" ");

}

**if**(n>=3){

s1=input.substring(0,3);

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

sb.append(s1).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

**Main**:

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

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

Scanner s= **new** Scanner(System.***in***);

**int** n,i;

System.***out***.println("enter the array size");

n=s.nextInt();

**int** array[]=**new** **int**[n];

**for**(i=0;i<n;i++){

array[i]=s.nextInt();

}

System.***out***.println(UserMainCode.*AverageElements*(array));

s.close();

}

}

**UserMainCode**:

**class** UserMainCode{

**public** **static** **double** AverageElements(**int** array[]){

**int** n, sum=0,count=0,count1=0;

**double** average;

n=array.length;

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

**for**(**int** j=1;j<n;j++){

**if**(i%j==0)

count++;

**if**(count==2){

sum= sum+array[i];

count1++;

}

}

}

average= sum/count1;

DecimalFormat df=**new** DecimalFormat("#.00");

**double** ddd=Double.*parseDouble*(df.format(average));

**return** ddd;

}

}

**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

**Main**:

import java.util.\*;

public class Main {

private static Scanner s ;

;

public static void main(String[] args) {

s = new Scanner (System.in);

int n = s.nextInt();

int a[] = new int[n];

int b[] = new int[n];

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

{

a[i] = s.nextInt();

}

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

{

b[i] = s.nextInt();

}

System.out.println(UserMainCode.sumCommonElements(a, b));

}

}

**UserMainCode**:

public class UserMainCode {

public static int sumCommonElements(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;

}

}

**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

**Main**:

**import** java.util.\*;

**public** **class** Main {

**private** **static** Scanner *s*;

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

*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();

              }

              System.***out***.println(UserMainCode.*getMiddleElement*(a));

       }

}

**UserMainCode**:

**public** **class** UserMainCode {

**public** **static** **int** getMiddleElement(**int** a[]){

**int** n = a.length;

**return** a[n/2];

       }

}

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

**Main:**

**import** java.util.\*;

**public** **class** Main {

**private** **static** Scanner *s*;

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

*s* = **new** Scanner(System.***in***);

      String s1 = *s*.next();

      System.***out***.println(UserMainCode.*getString*(s1));

       }

}

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** String getString(String s1){

              StringBuffer sb=**new** StringBuffer();

**char** a=s1.charAt(0);

**char** b=s1.charAt(1);

**if**(a!='j'&& b!='b')

              sb.append(s1.substring(2));

**else** **if**(a=='j' && b!='b')

              sb.append("j").append(s1.substring(2));

**else** **if**(a!='j' && b=='b')

              sb.append(s1.substring(1));

**else**

              sb.append(s1.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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

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

String s= sc.next();

**int** b = UserMainCode.*getvalues*(s);

**if**(b==1)

       System.***out***.println("Valid");

**else**

       System.***out***.println("Invalid");

}

}

**UserMainCode:**

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** UserMainCode {

**public** **static** **int** getvalues(String s) {

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

              {

              SimpleDateFormat sdf=**new** SimpleDateFormat("dd.MM.yyyy");

              sdf.setLenient(**false**);

**try**

              {

              Date d1=sdf.parse(s);

**return** 1;

              } **catch** (ParseException e) {

**return** -1;

              }

              }

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

              {

              SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");

              sdf.setLenient(**false**);

**try**

              {

              Date d1=sdf.parse(s);

**return** 1;

              } **catch** (ParseException e) {

**return** -1;

              }

              }

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

              {

              SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");

              sdf.setLenient(**false**);

**try**

              {

              Date d1=sdf.parse(s);

**return** 1;

              } **catch** (ParseException e) {

**return** -1;

              }

              }

**else**

**return** -1;

              }

}

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

**Main**:

**import** java.util.\*;

**import** java.util.Arrays;

**public** **class** Main {

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

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

**int** s = sc.nextInt();

**int** a[] = **new** **int**[s];

**for** (**int** i = 0; i < s; i++)

a[i] = sc.nextInt();

System.***out***.println(UserMainCode.*getBoundaryAverage*(a));

}

}

**UserMainCode**

**import** java.util.Arrays;

**public** **class** UserMainCode {

**public** **static** **float** getBoundaryAverage(**int** a[] ){

              Arrays.*sort*(a);

**int** sum = a[0] + a[a.length - 1];

**float** 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

**Main**:

 import java.util.\*;

public class Main {

private static Scanner s;

public static void main(String[] args) {

s = new Scanner(System.in);

String s1= s.next();

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

}

**UserMainCode**

public class UserMainCode{

public static int countVowels(String s1)

{

String s2=s1.toLowerCase();

String s3="aeiou";

int count=0;

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

{

for(int j=0;j<s3.length();j++)

{

if(s2.charAt(i)==s3.charAt(j))

{

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

**Main:**

import java.text.ParseException;

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));

sc.close();

}

}

**UserMainCode**:

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

public class UserMainCode {

public static String calculateBornDay(String s1) throws ParseException

{

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yy");

SimpleDateFormat sdf1=new SimpleDateFormat("MMMM");

Date d=sdf.parse(s1);

String s=sdf1.format(d);

return s.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 “reverseSubstring” 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

**Main**:

import java.util.\*;

public class Main {

private static Scanner s;

public static void main(String[] args) {

s =new Scanner(System.in);

   String input1= s.next();

int input2=s.nextInt();int input3=s.nextInt();

System.out.println(UserMainCode.retrieveString(input1,input2,input3));

}

}

**UserMainCode**

class UserMainCode {

public static String retrieveString(String input1, int input2, int input3) {

StringBuffer sb=new StringBuffer(input1);

sb.reverse();

String output=sb.substring(input2, input2+input3);

return output;

}

}

**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:                                 Sample Output 1:**

geniousRajKumarDev                         Yes

Raj

Dev

**Sample Input 2:                                 Sample Output 2:**

geniousRajKumarDev                         No

Dev

Raj

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** **int** stringFinder(String s1,String s2,String s3)

{

            String a1=s1.toLowerCase();

            String a2=s2.toLowerCase();

            String a3=s3.toLowerCase();

**if**(a1.contains(a2)&&a1.contains(a3))

            {

**if**(a1.indexOf(a2)<a1.indexOf(a3))

                        {

**return** 1;

                        }

**else**

**return** 2;

            }

**return** 0;

}}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

                        String s1=s.next();

                        String s2=s.next();

                        String s3=s.next();

**int** b=UserMainCode.*stringFinder*(s1, s2, s3);

**if**(b==1)

{

            System.***out***.println("yes");

}

**else**

            System.***out***.println("No");

s.close();

            }

}

**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 an 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:                                             Sample Output 1:**

265-265-7777                                                  Valid

**Sample Input 2:                                             Sample Output 2:**

265-65-7777                                                    Invalid

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** **int** validatePhoneNumber(String s1)

{

String s2 = s1.replaceAll("-", "");

**if** (s2.matches("[0-9]{10}"))

           {

**return** 1;

                                    }

**else**

**return** 2;

}

}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

                        String s1=s.nextLine();

**int** b=UserMainCode.*validatePhoneNumber*(s1);

**if**(b==1)

{

            System.***out***.println("Valid");

}

**else**

            System.***out***.println("Invalid");

s.close();

            }

}

**67. Month : Number of Days**

Given two inputs year and month (Month is coded as: Jan=0, Feb=1 ,Mar=2 ...), write aprogram 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 2integers as arguments and returns an integer. The first argument corresponds to the yearand the second argument corresponds to the month code. The method returns an integercorresponding 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 corresponds to the number of days in the month in the given year.

**Sample Input:                                                 Sample Output:**

2000                                                                29

1

**USERMAINCODE:**

**import** java.util.Calendar;

**public** **class** UserMainCode {

**public** **static** **int** getNumberOfDays(**int** y,**int** c)

{

            Calendar cal=Calendar.*getInstance*();

            cal.set(Calendar.***YEAR***, y);

            cal.set(Calendar.***MONTH***, c);

**int** day=cal.getActualMaximum(Calendar.***DAY\_OF\_MONTH***);

**return** day;

}

}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

**int** y=s.nextInt();

**int** c=s.nextInt();

System.***out***.println(UserMainCode.*getNumberOfDays*(y, c));

s.close();

}

}

**68. 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 preceded 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:                                 Sample Output 1:**

This is just a misconception                This is not just a misconception

**Sample Input 2:                                 Sample Output 2:**

Today is misty                                     Today is not misty

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** String negativeString(String s1)

{

            String str=s1.replace(" is ", " is not ");

**return** str;

}

}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

                        String s1=s.nextLine();

                        System.***out***.println(UserMainCode.*negativeString*(s1));

                        s.close();

            }

}

**69. 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:                                 Sample Output 1:**

-94923                                                 94923

**Sample Input 2:                                 Sample Output 2:**

-6t                                                        -1

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** String validateNumber(String s1)

{           String ss="-1";

**if** (s1.matches("[-]{1}[0-9]{1,}"))

            {

                        String st=s1.replace("-","");

**return** st;

            }

**else**

**return** ss;

                        }

}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

                        String s1=s.next();

                        System.***out***.println(UserMainCode.*validateNumber*(s1));

                        s.close();

            }

}

**70. 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:                                 Sample Output 1:**

717                                                      2

**Sample Input 2:                                 Sample Output 2:**

4534                                                    0

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** **int** countSeven(**int** n)

{

**int** rem,sum=0;

**while**(n>0)

            {

rem=n%10;

**if**(rem==7)

{

            sum++;

}

n=n/10;

            }

**return** sum;

}

}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

                        System.***out***.println(UserMainCode.*countSeven*(n));

                        s.close();

            }

}

**71. 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:**                                 **Sample Output 1:**

xxhixx                                                  hixxxx

**Sample Input 2:                                 Sample Output 2:**

XXxxtest                                               XXtestxx

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** String moveX(String s1)

{

            String s2="";

            String s3="";

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

            {

**char** c=s1.charAt(i);

**if**(c=='x')

                        {

s2=s2+s1.charAt(i);

}

**else**

                        {

s3=s3+s1.charAt(i)

                        }

            }

            String st=s3.concat(s2);

**return** st;

}

}

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

                        String s1=s.next();

                        System.***out***.println(UserMainCode.*moveX*(s1));

                        s.close();

            } }

**72. String Processing – IV**

Write a program to read a string and also a number N. Form a new string starting with 1stcharacter 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:                                 Sample Output 1:**

HelloWorld                                          HelWrd

**USERMAINCODE:**

**public** **class** UserMainCode {

**public** **static** String getStringUsingNthCharacter(String s1,**int** n)

{

            StringBuffer sb=**new** StringBuffer();

            sb.append(s1.charAt(0));

**for**(**int** i=n-1;i<s1.length();i+=n)

            {

                        sb.append(s1.charAt(i));

            }

**return** sb.toString();

} }

**MAIN:**

**import** java.util.\*;

**public** **class** Main {

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

                        Scanner s=**new** Scanner(System.***in***);

                        String s1=s.next();

**int** n=s.nextInt();

                        System.***out***.println(UserMainCode.*getStringUsingNthCharacter*(s1, n));

                        s.close();

            }

}

**73. 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

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **boolean** compareLastDigit(**int** c,**int** d)

{

**int** c1=c%10;

**int** d1=d%10;

**boolean** b=**false**;

**if**(c1==d1)

       {

       b=**true**;

       }

**return** b;

}

}

**Main**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

**int** c=s.nextInt();

**int** d=s.nextInt();

**boolean** res=UserMainCode.*compareLastDigit*(c,d);

**if**(res==**true**)

       {

              System.***out***.println("TRUE");

       }

**else**

       {

              System.***out***.println("FALSE");

       }

s.close();

       }

}

**74. 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

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **int** getDistinctSum(**int** a,**int** b,**int** c)

       {

**int** sum;

**if**(a==b)

              {

                     sum=c;

              }

**else** **if**(b==c)

              {

                     sum=a;

              }

**else** **if**(c==a)

              {

                     sum=b;

              }

**else**

              {

                     sum=a+b+c;

              }

**return** sum;

       }

}

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

**int** a=s.nextInt();

**int** b=s.nextInt();

**int** c=s.nextInt();

**int** sum=UserMainCode.*getDistinctSum*(a, b, c);

              System.***out***.println(sum);

              s.close();

       }

}

**75. 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

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **boolean** checkPattern(String str)

       {

       String str1=str.substring(0,3);

**int** a=0,b=0,c=0;

**char** c1=str1.charAt(0);

**char** c2=str1.charAt(1);

**char** c3=str1.charAt(2);

**boolean** b1=**false**;

**if**(Character.*isDigit*(c1)||Character.*isLetter*(c1))

       {

              a=1;

       }

**if**(c2=='i')

              {

                     b=1;

              }

**if**(c3=='x')

              {

                     c=1;

              }

**if**(a==1&&b==1&&c==1)

       {

              b1=**true**;

       }

**return** b1;

       }

}

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

              String str=s.nextLine();

**boolean** b2=UserMainCode.*checkPattern*(str);

**if**(b2==**true**)

              {

                     System.***out***.println("TRUE");

              }

**else**

              {

                     System.***out***.println("FALSE");

              }

s.close();

       }

}

**76. 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

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** String exchangeCharacters(String s1)

{

       String s2=s1.substring(1,s1.length()-1);

       StringBuffer sb=**new** StringBuffer();

**char** c1=s1.charAt(0);

**char** c2=s1.charAt(s1.length()-1);

       sb.append(c2).append(s2).append(c1);

**return** sb.toString();

}

}

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

       String s1=s.next();

       System.***out***.println(UserMainCode.*exchangeCharacters*(s1));

       s.close();

       }

}

**77. 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

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **boolean** validateString(String s1)

       {

**boolean** b=**false**;

**if**(s1.length()==3)

              {

**if**(s1.matches("[a-zA-z]{3}"))

              {

                     b=**true**;

              }

              }

**return** b;

       }

}

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

       String s1=s.next();

**boolean** b1=userMainCode.*validateString*(s1);

**if**(b1==**true**)

       {

              System.***out***.println("TRUE");

       }

**else**

       {

        System.***out***.println("FALSE");

       }

s.close();

       }

}

**78. 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

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** String repeatString(String s1,**int** n)

       {

              StringBuffer sb=**new** StringBuffer();

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

              {

                     sb.append(s1);

              }

**return** sb.toString();

       }

}

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

              String s1=s.next();

**int** n=s.nextInt();

              System.***out***.println(UserMainCode.*repeatString*(s1, n));

s.close();

       }

}

**79. 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

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **int** addOddNumbers(**int** n)

       {

**int** sum=0;

**for**(**int** i=1;i<=n;i+=2)

              {

                     sum=sum+i;

              }

**return** sum;

       }

}

**Main:**

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

       System.***out***.println(UserMainCode.*addOddNumbers*(n));

       s.close();

       }

}

**80. 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

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** String concatString(**int** n,String[] s1)

       {

              StringBuffer sb=**new** StringBuffer();

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

              {

                     sb.append(s1[i]).append(",");

              }

              String s2=sb.toString();

              String s3=s2.substring(0,s2.length()-1);

**return** s3;

       }

}

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

String s1[]=**new** String[n];

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

{

       s1[i]=s.next();

}

System.***out***.println(UserMainCode.*concatString*(n, s1));

s.close();

       }

}

**81.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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

       Scanner s=**new** Scanner(System.***in***);

**int** a=s.nextInt();

**int** b=s.nextInt();

**int** c=s.nextInt();

       System.***out***.println(UserMainCode.*calculateUnique*(a, b, c));

       s.close();

}

}

**UserMainCode**:

**public** **class** UserMainCode {

**public** **static** **int** calculateUnique(**int** a,**int** b,**int** c)

       {

**int** d=0;

**if**(a!=b&&a!=c&&b!=c)

       {

       d=3;

       }

**else** **if**(a==b&&a==c&&b==c)

       {

       d=1;

       }

**else** **if**((a!=b&&a==c&&b==c) || (a!=b&&a!=c&&b==c))

       {

       d=2;

       }

**else** **if**((a==b&&a!=c&&b==c) || (a==b&&a!=c&&b!=c))

       {

       d=2;

       }

**else** **if**((a==b&&a==c&&b!=c) || (a!=b&&a==c&&b!=c))

       {

              d=2;

              }

**return** d;

       }}

**82. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

**int** a=s.nextInt();

**int** b=s.nextInt();

**char** c = s.next().trim().charAt(0);

              System.***out***.println(UserMainCode.*calculator*(a, b, c));

              s.close();

       }}

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **int** calculator(**int** a,**int** b,**char** c)

       {

**int** a1=0;

**if**(c=='\*')

       {

       a1=a\*b;

       }

**else** **if**(c=='+')

       {

       a1=a+b;

       }

**else** **if**(c=='-')

       {

       a1=a-b;

       }

**else** **if**(c=='/')

       {

       a1=a/b;

       }

**else** **if**(c=='%')

       {

       a1=a%b;

       }

**return** a1;

       }}

**83. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main (String[] args)

       {

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

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

**int**[] arr = **new** **int**[n];

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

       arr[i] = sc.nextInt();

       }

       System.***out***.println(UserMainCode.*checkScores*(arr, n));

       sc.close();

       }

}

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **boolean** checkScores(**int** arr[], **int** n){

**boolean** b = **false**;

**for**(**int** i=0;i<n-1;i++){

**if**(arr[i] == 100){

**if**(arr[i+1] == 100){

              b = **true**;

**break**;

              }

              }

              }

**return** b;

              }

}

**84. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

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

**int** s=sc.nextInt();

**int** []a=**new** **int**[s];

**for**(**int** i=0;i<s;i++)

              {

              a[i]=sc.nextInt();

              }

**if**(UserMainCode.*scanArray*(a)==**true**)

System.***out***.println("TRUE");

**else**

System.***out***.println("FALSE");

              sc.close();

              }

}

**UserMainCode**:

**public** **class** UserMainCode {

**public** **static** **boolean** scanArray(**int**[] a)

       {

**int** u=0,l=0;

**boolean** b=**false**;

**if**(a.length>=4)

       l=4;

**else**

       l=a.length;

**for**(**int** i=0;i<l;i++)

**if**(a[i]==9)

       u=10;

**if**(u==10)

       b=**true**;

**return** b;

       }

     }

**85. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

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

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

              String[] str=**new** String[n];

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

              {

                     str[i]=sc.next();

              }

**int** c=sc.nextInt();

              System.***out***.println(UserMainCode.*countWord*(n,str,c));

              sc.close();

}}

**UserMainCode**:

**public** **class** UserMainCode {

**public** **static** **int** countWord(**int** n,String str[],**int** c)

       {

**int** count=0;

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

       {

**if**(str[i].length()==c)

       {

       count++;

       }

       }

**return** count;

       }

}

**86. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main (String[] args)

       {

       Scanner s=**new** Scanner(System.***in***);

**int** a=s.nextInt();

**int** b=s.nextInt();

**int** c=s.nextInt();

**int** d=s.nextInt();

System.***out***.println(UserMainCode.*findDistance*(a,b,c,d));

s.close();

}

}

**UserMainCode**:

**public** **class** UserMainCode {

**public** **static** **int** findDistance(**int** a,**int** b,**int** c,**int** d) {

**long** q=(**int**)Math.*round*(Math.*sqrt*(((a-c)\*(a-c))+((b-d)\*(b-d))));

**return** (**int**) q;

              }

              }

**87. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

       String s1=s.nextLine();

       UserMainCode.*countWord*(s1);

       s.close();

       }

       }

**UserMainCode**:

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** **void** countWord(String s1){

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

**int** n=st.countTokens();

       System.***out***.println(n);

}

}

**88. 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

**Main**:

**import** java.util.\*;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

**int** a=s.nextInt();

**int** b=s.nextInt();

**int** c=s.nextInt();

              System.***out***.println(UserMainCode.*getSumMaxMin*(a,b,c));

              s.close();

}}

**UserMainCode**:

**public** **class** UserMainCode {

**public** **static** **int** getSumMaxMin(**int** a,**int** b,**int** c)

       {

**int** d=0;

**if**(a<b&&b<c)

       {

       d=a+c;

       }

**else** **if**(a<b&&b>c)

       {

       d=b+c;

       }

**else** **if**(a>b&&b<c)

       {

       d=a+b;

       }

**return** d;

       }}

**89. 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

**MAIN**

**import** java.util.Scanner;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

System.***out***.println(UserMainCode.*convertDecimalToBinary*(n));

s.close();

}

}

**UserMainCode**

**public** **class** UserMainCode {

**public** **static** **long** convertDecimalToBinary(**int** n){

       String s1=Integer.*toBinaryString*(n);

**long** y=Long.*parseLong*(s1);

**return** y;

}

}

**90.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

**MAIN**

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** **void** main(String[] args)

{

       Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

**int** n1=s.nextInt();

System.***out***.println(UserMainCode.*returnLastRepeatedCharacters*(s1,n1));

s.close();

}

}

**USERMAINCODE**

**public** **class** UserMainCode{

**public** **static** String returnLastRepeatedCharacters(String s1, **int** n1)

{

StringBuffer sb = **new** StringBuffer();

**for**(**int** i = 0 ; i < n1 ; i++)

sb.append(s1.substring(s1.length()-n1, s1.length()));

**return** sb.toString();

}

}

**91.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

**MAIN**

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args)

       {

              Scanner s=**new** Scanner(System.***in***);

       String s1=s.nextLine();

**if**(UserMainCode.*validateString*(s1)==**true**) System.***out***.println("TRUE");

**else** System.***out***.println("FALSE");

       s.close();

       }

}

**USERMAINCODE**

**public** **class** UserMainCode {

**public** **static** **boolean** validateString(String s)

       {

**boolean** b=**false**;

**if**(s.charAt(0)=='0'||s.charAt(0)=='1'||s.charAt(0)=='2'||s.charAt(0)=='3'||s.charAt(0)=='4'||s.charAt(0)=='5'||s.charAt(0)=='6'||s.charAt(0)=='7'||s.charAt(0)=='8'||s.charAt(0)=='9'){

                     b=**false**;

              }

**else**

                     b=**true**;

**return** b;

       }

}

**92.String 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

**MAIN**

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** **void** main(String[] args)

{

       Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

System.***out***.println(UserMainCode.*getAlternateChars*(s1));

s.close();

}

}

**USERMAINCODE**

**public** **class** UserMainCode{

**public** **static** String getAlternateChars(String s)

{

//String s1=s.replaceAll(“ “, “”);

StringBuffer sbf = **new** StringBuffer();

**for**(**int** i = 0; i < s.length() ; i=i+2)

{

sbf.append(s.charAt(i));

}

String str = sbf.toString();

**return** str;

}

}

**93. 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](mailto:admin@xyz.com)

**Sample Output 1:**

admin

**MAIN CLASS**

**import** java.util.Scanner;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

System.***out***.println(UserMainCode.*fetchUserName*(s1));;

s.close();

}

}

**USERMAINCODE**

**import** java.util.StringTokenizer;

**public** **class** UserMainCode {

**public** **static** String fetchUserName(String s1) {

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

              String s2=st.nextToken();

**return**(s2);

              }}

**94. 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

**MAIN**

**import java.util.Scanner;**

**public class Main {**

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

**Scanner s=new Scanner(System.*in*);**

**String s1=s.nextLine();**

**String s2=s.nextLine();**

**int n=s.nextInt();**

**boolean output=UserMainCode.*isEqual*(s1,s2,n);**

**System.*out*.println(output);**

**s.close();**

**}**

**}**

**USERMAINCODE**

**public class UserMainCode {**

**public static boolean isEqual(String s1,String s2,int n){**

**boolean a=false;**

**if(n<s1.length()&&n<s2.length())**

**{**

**char c=s1.charAt(n);**

**char d=s2.charAt(s2.length()-n);**

**String s3=Character.*toString*(c);**

**//System.out.println(s3);**

**String s4=Character.*toString*(d);**

**//System.out.println(s4);**

**if(s3.equalsIgnoreCase(s4))**

**{**

**a=true;**

**}**

**else**

**{**

**a=false;**

**}**

**}**

**return a;**

**}**

**}**

**95. 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

**MAIN CLASS**

**import** java.util.Scanner;

**public** **class** Main{

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

Scanner s=**new** Scanner(System.***in***);

**int** m=s.nextInt();

**int**[] n1=**new** **int**[m];

**for**(**int** i=0;i<m;i++){

              n1[i]=s.nextInt();

       }

System.***out***.println(UserMainCode.*checkDifference*(n1));

s.close();

}

}

**USERMAIN CODE**

**public** **class** UserMainCode {

**public** **static** **int** checkDifference(**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;

}

}

**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!

**MAIN CLASS**

**import** java.util.Scanner;

**public** **class** Main {

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

Scanner s=**new** Scanner(System.***in***);

String s1=s.nextLine();

System.***out***.println(UserMainCode.*printCapitalized*(s1));

s.close();

}

}

**USERMAIN CODE**

**import** java.util.StringTokenizer;

**public** **class** UserMainCode{

**public** **static** String printCapitalized(String s1){

StringBuffer sb=**new** StringBuffer();

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

**while**(t.hasMoreTokens()){

String s2=t.nextToken();

String s3=s2.substring(0,1);

String s4=s2.substring(1, s2.length());

sb.append(s3.toUpperCase()).append(s4).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.]

**Main:**

**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**[20];

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

       {

       a[i]=s.nextInt();

       }

**int** max=UserMainCode.*findMaxDistance*(a);

              System.***out***.println(max);

s.close();

       }

}

**UserMainCode:**

**public** **class** UserMainCode {

**static** **int** findMaxDistance(**int**[] a)

       {

**int** max=0,index=0;

**for**(**int** i=0;i<19;i++)

       {

**int** d=Math.*abs*(a[i]-a[i+1]);

**if**(d>max)

       {

       max=d;

**if**(a[i]>a[i+1])

       {

       index=i;

       }

**else**

       {

       index=i+1;

       }

       }

}

**return** index;

       }

       }

**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)**

**Main:**

**import** java.util.Scanner;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

**int** n1=s.nextInt();

**int** n2=s.nextInt();

              System.***out***.println(UserMainCode.*addPalindromes*(n1,n2));

              s.close();

              }

              }

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **int** addPalindromes(**int** n1,**int** n2){

**int** sum=0;

**for**(**int** i=n1;i<=n2;i++){

**int** r=0,n3=i;

**while**(n3!=0){

       r=(r\*10)+(n3%10);

       n3=n3/10;

       }

**if**(r==i)

       sum=sum+i;

       }

**return** sum;

       }

}

**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

**Main:**

**import** java.util.Scanner;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

              String s1=s.nextLine();

       UserMainCode.*validatePAN*(s1);

       s.close();

}

}

**UserMainCode:**

**public** **class** UserMainCode {**public** **static** **void** validatePAN(String s1) {

**if**(s1.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))

       {

       System.***out***.println("Valid");

       }

**else**

              System.***out***.println("Invalid");

              }

              }

**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]**

**Main:**

**import** java.util.Scanner;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

**int** n=s.nextInt();

              System.***out***.println(UserMainCode.*getSumOfNfibos*(n));

              s.close();

              }

              }

**UserMainCode:**

**public class UserMainCode {**

**public static int getSumOfNfibos(int n){**

**int a=-1,b=1,c=0,d=0;**

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

**{**

**c=a+b;**

**d=d+c;**

**a=b;**

**b=c;**

**}**

**return d;**

**}**

**}**

**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

**Main:**

**import** java.util.Scanner;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

              String s1=s.nextLine();

**int** b=UserMainCode.*testVowels*(s1);

**if**(b==1)

                     System.***out***.println("Yes");

**else**

                           System.***out***.println("No");

                     s.close();

       }

              }

**UserMainCode:**

**public** **class** UserMainCode {

**public** **static** **int** testVowels(String s1) {

**int** b;

**int** n1=0,n2=0,n3=0,n4=0,n5=0;

              String s2=s1.toLowerCase();

**for**(**int** i=0;i<s2.length();i++){

**char** c=s2.charAt(i);

**if**(c=='a')

              n1++;

**if**(c=='e')

              n2++;

**if**(c=='i')

              n3++;

**if**(c=='o')

              n4++;

**if**(c=='u')

              n5++;}

**if**(n1==1&&n2==1&n3==1&&n4==1&&n5==1)

              b=1;

**else** b=2;

**return** b;

              }

       }

**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

**import** java.util.Scanner;

**public** **class** Main {

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

              Scanner s=**new** Scanner(System.***in***);

              String s1=s.nextLine();

              String s2=s.nextLine();

**int** p=UserMainCode.*compareDashes*(s1,s2);

**if**(p==1)

       System.***out***.println("Yes");

**else**

              System.***out***.println("No");

       s.close();

       }

       }

**import** java.util.ArrayList;

**public** **class** UserMainCode {

**public** **static** **int** compareDashes(String s1, String s2) {

       ArrayList<Integer>l1=**new** ArrayList<Integer>();

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

       {

**if**(s1.charAt(i)=='-')

       {

       l1.add(i);

       }

       }

       ArrayList<Integer>l2=**new** ArrayList<Integer>();

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

       {

**if**(s2.charAt(i)=='-')

       {

       l2.add(i);

       }

       }

       //System.out.println(l1);

       //System.out.println(l2);

**if**(l1.equals(l2))

       {

**return** 1;

       }

**else**

**return** 2;

       }

       }

**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

**import java.util.Scanner;**

**public class Main {**

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

**Scanner s=new Scanner(System.*in*);**

**String s1=s.nextLine();**

**String s2=s.next();**

**System.*out*.println(UserMainCode.*reShape*(s1,s2));**

**s.close();**

**}**

**}**

**public class UserMainCode {**

**public static String reShape(String s,String s1){**

**StringBuffer sb=new StringBuffer(s);**

**StringBuffer sb2=new StringBuffer();**

**String s2=sb.reverse().toString();**

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

**{**

**sb2.append(s2.charAt(i));**

**sb2.append(s1);**

**}**

**sb2.deleteCharAt(sb2.length()-1);**

**//System.out.println(sb2.toString());**

**return sb2.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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int i;

int n = sc.nextInt();

int a[] = new int[n];

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

                a[i] = sc.nextInt();

                }

UserMainCode.removeTens(a);

sc.close();

}

}

**import** java.util.\*;

**public** **class** UserMainCode {

**public** **static** **void** removeTens(**int** a[]){

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

**int** i,k = 0;

**int** b[] = **new** **int**[a.length];

ArrayList<Integer> al = **new** ArrayList<Integer>();

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

**if** (a[i] != 10) {

al.add(a[i]);

}

}

**if** (al.size() < a.length) {

k = a.length- al.size();

**for** (i = 0; i < k; i++) {

al.add(0);

}

}

**int** b1[] = **new** **int**[a.length];

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

b1[i] = al.get(i);

System.***out***.println(b1[i]);

}

}}

|  |  |
| --- | --- |
| |  | | --- | | **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 $. Print the final string.  Include a class **UserMainCode** with a static method **getLastLetter** which accepts a string.  The return type (string) should return the final string.  Create a Class Main which would be used to read a string, and call the static method present  in UserMainCode.  **Input and Output Format:**  Input consists of a string.  Output consists of a string (the final string).  Refer sample output for formatting specifications.  **Smaple Input :**  This is a cat  **Sample Output :**  S$S$A$T  **Main:**  **public** **class** Main {    **public** **static** **void** main(String[] args) {                Scanner s=**new** Scanner(System.***in***);                String input=s.nextLine();                System.***out***.println(UserMainCode.*getLastLetter*(input));           }    }  **UserMainCode:**  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** String getLastLetter(String input){                String str1=**null**;                StringTokenizer st=**new** StringTokenizer(input," ");                StringBuffer sb=**new** StringBuffer();  **while**(st.hasMoreTokens()){                str1=st.nextToken();                /// String str2=Character.toString(str1.charAt(str1.length()-1));                String str2=str1.substring(str1.length()-1);                String str3= str2.toUpperCase();                sb.append(str3).append("$");                }sb.deleteCharAt(sb.length()-1);  **return** sb.toString();                }                }  **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      **Main:**    **import** java.util.\*;  **public** **class** Main {         /\*\*         \* **@param** args         \*/  **public** **static** **void** main(String[] args) {         // **TODO** Auto-generated method stub         HashMap<Integer, String>hm=**new** HashMap<Integer, String>();         Scanner s=**new** Scanner(System.***in***);  **int** n=s.nextInt();  **for**(**int** i=0;i<n;i++)         {  **int** a=s.nextInt();                String s1=s.next();                hm.put(a,s1);           }         System.***out***.println(UserMainode.*getMaxKeyValue*(hm));           }  }      **Hashmap:**  **import** java.util.\*;  **import** java.util.HashMap;  **import** java.util.Iterator;  **public** **class** UserMainode {  **public** **static** String getMaxKeyValue(HashMap<Integer, String> hm) {  **int** max=0;                String s3=**null**;                Iterator<Integer>itr=hm.keySet().iterator();  **while**(itr.hasNext())                {  **int** b=itr.next();  **if**(b>max)                {  max=b;  s3=hm.get(b);  }  }  **return** (s3);  }  }  **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      **Main:**  **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner s = new Scanner(System.in);**  **int n = s.nextInt();**  **String[] s1 = new String[n];**  **for(int i=0;i<n;i++){**  **s1[i] = s.next();**  **}**  **int out=(userMainCode.validateNumber(s1));**  **System.out.println(out);**    **}**  **}**  **UserMainCode:**  **class userMainCode{**  **public static int validateNumber(String[] s1){**  **int b =0 ,count,out=0;**  **for(int i=0;i<s1.length;i++){**  **String s2 = s1[i];**  **if(s2.matches("[0-9.]{1,}"))**    **{     count =0;**  **for(int j=0;j<s2.length();j++)**  **{**  **char c = s2.charAt(j);**  **if(c=='.')**  **count++;**  **}**  **if(count>1)**  **b=1;**  **}**  **else**  **b=1;**  **}**  **if(b==0){**  **out=1;**  **}**  **else out=-1;**  **return out;**  **}**  **}**    **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      **Main:**  **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.*getDay*(s1));         }                  // **TODO** Auto-generated method stub           }  **UserMainCode:**  **import** java.text.SimpleDateFormat;  **import** java.text.ParseException;  **import** java.util.Date;  **public** **class** UserMainCode {    **public** **static** String getDay(String s1) **throws** ParseException  {  SimpleDateFormat sdf=**new** SimpleDateFormat("MM-dd-yyyy");  SimpleDateFormat sdf1=**new** SimpleDateFormat("EEEEE");  Date d=sdf.parse(s1);  String s=sdf1.format(d);  **return** s;  }  }  **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\  **Main:**  **import** java.util.\*;      **public** **class** Main {  **public** **static** **void** main(String[] args){           Scanner sc=**new** Scanner(System.***in***);         String input1=sc.next();         String input2=sc.next();         System.***out***.println(UserMainCode.*extractMax*(input1,input2));    }  }    **Usermaincode:**    **import** java.util.StringTokenizer;  **import** java.util.\*;    **public** **class** UserMainCode {  **public** **static** String extractMax(String input1,String input2){  **int** max=0;                String s3=**null**;                StringTokenizer st=**new** StringTokenizer(input1,"-");  **while**( st.hasMoreTokens())                {                String s2=st.nextToken();  **int** n=s2.length();  **if**(n>max)                {                max=n;                s3=s2;                }                }  **return**(s3);                }}      **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$punjab    **Main:**  **import** java.util.HashMap;  **import** java.util.Scanner;  **public** **class** Main {    **public** **static** **void** main(String[] args) {                 Scanner sc=**new** Scanner(System.***in***);  **int** n=sc.nextInt();                HashMap<String,String> hm=**new**                HashMap<String,String>();  **for**(**int** i=0;i<n;i++)                {                String s1=sc.next();                String s2=sc.next();                hm.put(s1,s2);                }                String sa=sc.next();                System.***out***.print(UserMainCode.*getCapital*(hm,sa));                }             }    **UserMainCode:**  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.Map;  **public** **class** UserMainCode {  **public** **static** String getCapital(HashMap<String,String>         hm,String sa)         {         String chan=**null**;         Iterator<String>it=hm.keySet().iterator();         StringBuffer sb=**new** StringBuffer();  **while**(it.hasNext()){         String a=it.next();  **if**(a.equals(sa))         {         chan=hm.get(a);         sb.append(chan).append("$").append(sa);         }         }  **return** sb.toString();         }         }    **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    **Main:**  **import** java.util.\*;  **public** **class** Main {    **public** **static** **void** main(String[] args) {                Scanner sc=**new** Scanner(System.***in***);                String inpList=sc.nextLine();                System.***out***.println(UserMainCode.*calculateWordSum*(inpList));                }             }  **UserMainCode:**  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** **int** calculateWordSum(String inp) {    **int** count=0;         String st[]=inp.split(" ");         String s1=st[0];         String slst=st[st.length-1];  **if**(s1.equals(slst))         {         count=s1.length();         }  **else**         {                count=s1.length()+slst.length();                }  **return** count;                }                }      **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    **Main:**  **package** vowels;  **import** java.util.\*;    **public** **class** Main {    **public** **static** **void** main(String[] args) {      **int** n;                       Scanner sc=**new** Scanner(System.***in***);                       n=Integer.*parseInt*(sc.nextLine());                       String[] str=**new** String[n];  **for**(**int** i=0;i<n;i++)                       {                       str[i]=sc.nextLine();                       }                       ArrayList<String> arr=**new** ArrayList<String>();                       arr=UserMainCode.*matchCharacter*(str);                       Iterator<String> it=arr.iterator();  **while**(it.hasNext())                       {                       System.***out***.println(it.next());                       }             }  }    **Usermaincode:**  **package** vowels;  **import** java.util.\*;  **public** **class** UserMainCode {    **public** **static** ArrayList<String> matchCharacter (String[] ss)                {                ArrayList<String> as=**new** ArrayList<String>();  **for**(**int** i=0;i<ss.length;i++)                {                String sp=ss[i];  **char**[] mp=sp.toLowerCase().toCharArray();  **if**((mp[0]=='a'||mp[0]=='e'||mp[0]=='i'||mp[0]=='o'||mp[0]=='u')&&(mp[sp.length()-                1]=='a'||mp[sp.length()-1]=='e'||mp[sp.length()-1]=='i'||mp[sp.length()-                1]=='o'||mp[sp.length()-1]=='u'))                {                as.add(sp);                }                }  **return** as;                }                } |   **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  **main:**  **import** java.util.\*;  **import** java.text.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                HashMap<Integer,String> hm1=**new** HashMap<Integer,String>();  **int** n;                Scanner sc=**new** Scanner(System.***in***);                n=Integer.*parseInt*(sc.nextLine());  **for**(**int** i=0;i<n;i++)                {                hm1.put(Integer.*parseInt*(sc.nextLine()),sc.nextLine());                }                ArrayList<String> al1=**new** ArrayList<String>();                al1=UserMainCode.*getName*(hm1);                Iterator<String> it=al1.iterator();  **while**(it.hasNext())                {                System.***out***.println(it.next());         }    }  }    **Usermaincode:**      **import** java.util.\*;  **import** java.text.\*;  **public** **class** UserMainCode {  **public** **static** ArrayList<String> getName(HashMap<Integer,String> hm1)         {         ArrayList<String> al2=**new** ArrayList<String>();         Iterator<Integer> it =hm1.keySet().iterator();  **while**(it.hasNext())         {  **int** id=it.next();         String name=hm1.get(id);  **if**(name.matches("[a-z]{1,}.\*[0-9]{1,}.\*[A-Z]{1}"))         al2.add(name);         }  **return** al2;         }         }  **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  **USERMAINCODE:**  **import java.util.ArrayList;**  **public class UserMainCode**  **{**  **public static int year (ArrayList<Integer> a1)**  **{**  **int max=0,pos=0;**  **for(int i=1;i<a1.size();i+=2)**  **{**  **if(a1.get(i)>max)**  **{**  **max=a1.get(i);**  **pos=i;**  **}**  **}**  **return a1.get(pos-1);**  **}**  **}**  **MAIN:**  **import java.util.\*;**  **class Main**  **{**  **public static void main(String [] args)**  **{**  **Scanner s=new Scanner(System.*in*);**  **ArrayList<Integer> a1=new ArrayList<Integer>();**  **int n=s.nextInt();**  **n=n\*2;**  **for(int i=0;i<n;i++)**  **{**  **a1.add(s.nextInt());**  **}**  **System.*out*.println(UserMainCode.*year*(a1));**  **s.close();**  **}**  **}**    **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  **Main:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                {                Scanner s=**new** Scanner(System.***in***);  **int** n=s.nextInt();                System.***out***.println(UserMainCode.*addNumbers*(n));                }                       }                }  **Usermaincode:**    **public** **class** UserMainCode {  **public** **static** **int** addNumbers(**int** n) {  **int** sum=0;**int** k=0;**int** sum1=0;  **for**(**int** i=1; i<=n; i++)                { k=0;  **for**(**int** j=1; j<=i; j++)                {  **if**(i%j==0)                k++;                }  **if**(k!=2)                {                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  **Main:**  import java.util.\*;  import java.text.\*;  public class Main {  public static void main(String[] args) {                Scanner s=new Scanner(System.in);                String s1=s.next();                UserMainCode.convertDateFormate(s1);                }           }    **Usermaincode:**  import java.util.\*;  import java.text.\*;  public class UserMainCode {         public static void convertDateFormate(String s1) {                SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");                sdf.setLenient(false);                try {                Date d1=sdf.parse(s1);                SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yy");                String s2=sdf1.format(d1);                System.out.println(s2);                } catch (ParseException e) {                e.printStackTrace();                }                }                }  **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  **UserMainCode:**  **public class UserMainCode**  **{**  **public static int dateformat(String s1) throws ParseException**  **{**  **String s2=" ";**  **int n=-1;**  **if(s1.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))**  **{**  **SimpleDateFormat sdf=new SimpleDateFormat("dd.MM.yyyy");**  **Date d=sdf.parse(s1);**  **s2=sdf.format(d);**  **n=1;**  **}**  **else if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{2}"))**  **{**  **SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yy");**  **Date d1=sdf1.parse(s1);**  **s2=sdf1.format(d1);**  **n=1;**  **}**  **else if(s1.matches("[0-9]{2}[-]{1}[0-9]{2}[-]{1}[0-9]{4}"))**  **{**  **SimpleDateFormat sdf2=new SimpleDateFormat("dd-MM-yyyy");**  **Date d2=sdf2.parse(s1);**  **s2=sdf2.format(d2);**  **n=1;**  **}**  **else**  **{**  **n=-1;**  **}**  **return n;**  **}**  **}**  **MAIN:**  **import java.text.ParseException;**  **import java.util.\*;**  **class Main**  **{**  **public static void main(String [] args) throws ParseException**  **{**  **Scanner s=new Scanner(System.*in*);**  **String s1=s.next();**  **int b=UserMainCode.*dateformat*(s1);**  **if(b==1)**  **{**  **System.*out*.println("Valid");**  **}**  **Else**  **{**  **System.*out*.println("Invalid");**  **}**  **s.close();**  **}**  **}**  **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    **Main:**  import java.util.\*;  import java.text.\*;  public class Main {              public static void main(String[] args) {                                      Scanner sc=new Scanner(System.in);                                      String s=sc.next();                                      System.out.println(UserMainCode.convertFormate(s));                                      }                }    **Usermaincode:**  import java.util.\*;  import java.text.\*;  public class UserMainCode {              public static String convertFormate(String s) {                          StringTokenizer t=new StringTokenizer(s,"-");                          String s1=t.nextToken();                          String s2=t.nextToken();                          String s3=t.nextToken();                          StringBuffer sb=new StringBuffer();                          sb.append(s1.substring(0, s1.length()-1)).append('-');                          sb.append(s1.charAt(s1.length()-1)).append(s2.charAt(0)).append('-');                          sb.append(s2.substring(1, s2.length())).append(s3.charAt(0)).append('-');                          sb.append(s3.substring(1, s3.length()));                          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  **Main:**  **import** java.util.\*;      **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();         }  **int** b=sc.nextInt();  System.***out***.println(UserMainCode.*addAndReverse*(n,b,a)) ;  sc.close();  }  }  **Usermaincode:**  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** **int** addAndReverse(**int** n,**int** b,**int** a[])         {      **int** i=0,sum=0,r=0;  **for**(i=0;i<a.length;i++)         {  **if**(a[i]>b)         {         sum=sum+a[i];         }         }         System.***out***.println(sum);  **while**(sum!=0)         {         r=((r\*10)+(sum%10));         sum=sum/10;         }  **return** r;  }  }  **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  **Main:**  **import** java.util.\*;  **import** java.text.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {         Scanner sc=**new** Scanner(System.***in***);  String s1=sc.next();  UserMainCode u=**new** UserMainCode();  {         System.***out***.println(u.nextYearDay(s1));  }  }  }  **Usercodemain:**  **import** java.util.\*;  **import** java.text.\*;  **public** **class** UserMainCode  {  **public** String nextYearDay(String s1)         {                String s=**null**;                SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");                sdf.setLenient(**false**);  **try** {                Date d1=sdf.parse(s1);                Calendar cal=Calendar.*getInstance*();                cal.setTime(d1);                cal.add(Calendar.***YEAR***, 1);                Date d2=cal.getTime();                SimpleDateFormat sdf1=**new** SimpleDateFormat("EEEEE");                s=sdf1.format(d2);                }  **catch** (ParseException e)                {                e.printStackTrace();                }  **return** s;                }              }  **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  **Main:-**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                Scanner s=**new** Scanner(System.***in***);  **int** n=s.nextInt();                       UserMainCode.*getSumOfSquaresOfDigits*(n);                       s.close();                         }         }    **UserMainCode:-**  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** **void** getSumOfSquaresOfDigits(**int** n) {  **int** a=n;  **int** rem=0;  **int** sum=0;  **while**(a!=0)                {                rem=a%10;                sum=sum+(rem\*rem);                a=a/10;                }                System.***out***.println(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      **Main:-**    **import** java.util.Scanner;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  Scanner sc = **new** Scanner(System.***in***);  **int** n = sc.nextInt();  UserMainCode.*sumOfOddEvenPositioned*(n);  sc.close();  }  }    **UserMainCode:-**    **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** **void** sumOfOddEvenPositioned(**int** n) {  **int** rem = 0, i = 0;  **int** a[] = **new** **int**[10];  **while** (n > 0) {  rem = n % 10;  a[i] = rem;  n = n / 10;  i++;  }  **int** sume = 0, sumo = 0;  **for** (**int** j = i - 1; j >= 0; j--) {  **if**(j%2!=0)  {  sumo = sumo + a[j];  }  **else**  {  sume = sume + a[j];  }  }  **if** (sume == sumo) {  System.***out***.println("Yes");  } **else**  System.***out***.println("No");  }  }        **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    **Main:-**    **import** java.util.ArrayList;  **import** java.util.Iterator;  **import** java.util.Scanner;  **public** **class** Main{  **public** **static** **void** main(String []args){  Scanner sc=**new** Scanner(System.***in***);  ArrayList<Integer> al=**new** ArrayList<Integer>();  ArrayList<Integer> al1=**new** ArrayList<Integer>();  **int** n=Integer.*parseInt*(sc.nextLine());  **for**(**int** i=0;i<n;i++)  {  al.add(sc.nextInt());  }  al1=UserMainCode.*removeMultiplesOfThree*(al);  Iterator it=al1.iterator();  **while**(it.hasNext())  {  System.***out***.println(it.next());  }  }  }    **UserMainCode:-**    **import** java.util.ArrayList;  **import** java.util.Iterator;  **import** java.util.StringTokenizer;  **public** **class** UserMainCode  {  **public** **static** ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> al)  {  ArrayList<Integer> al2=**new** ArrayList<Integer>();  **for**(**int** i=0;i<al.size();i++)  {  **if**((i+1)%3!=0)  al2.add(al.get(i));  }  **return** al2;  }  }    **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    **Main:-**    **MAIN**  **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*.println(UserMainCode.*getSubstring*(s, s1));**  **sc.close();**  **}**  **}**  **USERMAINCODE**  **public class UserMainCode{**  **public static int getSubstring(String s,String s1){**  **int t=s1.length();**  **int count=0;**  **for(int i=0;i<s.length()-t+1;i++)**  **{**  **String s3=s.substring(i,t+i);**  **if(s3.equals(s1))**  **{**  **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        Main:-  **import java.util.Scanner;**  **public class Main{**  **public static void** main(String[] args){  Scanner s=**new** Scanner(System.***in***);  int a=s.nextInt();  int b=s.nextInt();  int c=s.nextInt();  System.***out***.println(UserMainCode.*luckySum*(a,b,c));  }  }  UserMainCode:-  **public class** UserMainCode{  **public static int** luckySum(**int** a, **int** b, **int** c)  {  if(a == 13)  **return** 0;  if(b == 13)  return a;  if(c == 13)  return (a + b);  return (a + b + c);  }  }    **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  **Main:-**  **import** java.util.Scanner;  **public** **class** Main {  **public** **static** **void** main(String[] args)  {  **int** n;  Scanner sc=**new** Scanner(System.***in***);  n=sc.nextInt();  **int**[] a=**new** **int**[n];  **for**(**int** i=0;i<n;i++)  {  a[i]=sc.nextInt();  }  **boolean** s=UserMainCode.*checkTripplets*(a);  **if**(s==true)  System.***out***.println("TRUE");  **else**  System.***out***.println("FALSE");  }  }    **UserMainCode:-**  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** **boolean** checkTripplets(**int**[] a)  {  **boolean** b=**false**;    **for**(**int** i=0;i<a.length-2;i++)  {  **if**((a[i]==a[i+1])&&(a[i+1]==a[i+2]))  {  b=**true**;  }  }  **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  **Main:-**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String []args){  Scanner sc=**new** Scanner(System.***in***);  String s=sc.nextLine();  **int** n=Integer.*parseInt*(sc.nextLine());  System.***out***.println(UserMainCode.*repeatFirstThreeCharacters*(s,n));  sc.close();  }  }    **UserMaincode:-**    **import** java.util.\*;  **public** **class** UserMainCode  {  **public** **static** String repeatFirstThreeCharacters(String s,**int** n)  {  StringBuffer sb=**new** StringBuffer();  StringBuffer sb1=**new** StringBuffer();  **if**(s.length()>3)  { sb.append(s.substring(0,3));  s=sb.toString();  }  **for**(**int** i=0;i<n;i++)  sb1.append(s);  **return** sb1.toString();  }  }    **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  **Main:-**  **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String[] args)  {  **int** n;  Scanner sin = **new** Scanner(System.***in***);  n = sin.nextInt();  String[] a1 = **new** String[n];  **for**(**int** i=0;i<n;i++)  {  a1[i] = sin.next();  }  a1 = UserMainCode.*orderElements*(a1);  **for**(**int** i=0;i<a1.length;i++)  System.***out***.println(""+a1[i]);  }  }    **UserMainCode:-**    **import** java.util.\*;  **public** **class** UserMainCode  {  **public** **static** String[] orderElements(String[] arr)  {  HashSet<String> al=**new** HashSet<String>();  **for**(**int** i=0;i<arr.length;i++)  {  al.add(arr[i]);  }  Iterator<String> itr=al.iterator();  String ar[] = **new** String[al.size()];  **int** i =0 ;  **while**(itr.hasNext()){  ar[i] = itr.next();  i++;  }  Arrays.*sort*(ar);  **return** ar;  }  }    **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  Main:  **import** java.util.\*;  **public** **class** Main  {    **public** **static** **void** main(String[] args)         {                Scanner sc = **new** Scanner(System.***in***);         String s = sc.next();         System.***out***.println(UserMainCode.*CheckID*(s));         sc.close();  }}  UserMainCode:  **public** **class** UserMainCode  {  **public** **static** **boolean** CheckID(String s)  {  **boolean** b=**false**;  **if**(s.matches("(CPT)[-]{1}[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$  Main:  **import** java.util.Scanner;  **public** **class** Main  {  **public** **static** **void** main(String[] arg)  {  Scanner s=**new** Scanner(System.***in***);  **int** n=Integer.*parseInt*(s.nextLine());  String[] sc=**new** String[n];  **for**(**int** i=0;i<n;i++)  {  sc[i]=s.nextLine();  }  **int** a=Integer.*parseInt*(s.nextLine());  System.***out***.println(UserMainCode.*formString*(n,sc,a));  s.close();  }  }  UserMainCode:  **public** **class** UserMainCode {  **public** **static** String formString(**int** n,String[] input,**int** a)         {         StringBuffer sb=**new** StringBuffer();  **for**(**int** i=0;i<n;i++)         {  **if**(input[i].length()>=a)         {         String a1=input[i];         sb.append(a1.charAt(a-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  S.36) **import** java.util.Scanner;  **public** **class** Main {  **public** **static** **void** main(String []args){  Scanner sc=**new** Scanner(System.***in***);  String n=sc.nextLine();  System.***out***.println(UserMainCode.*validate*(n));  sc.close();  }  }  **public** **class** UserMainCode  {  **public** **static** String validate(String s)  {  String w="FALSE";  **if**(s.length()==4 &&  (Character.*isDigit*(s.charAt(0))||Character.*isAlphabetic*(s.charAt(0)))&&s.charAt(1)=='R')  {  **if**(Character.*isDigit*(s.charAt(2)))  w="TRUE";  }  **return** w;  }  }  **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  AcB/TRUE  Main:  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[]args){         Scanner s=**new** Scanner(System.***in***);                  //Regular Expression – 2 (Age Validator) pg.No:150         String n=s.nextLine();  **boolean** b=UserMainCode.*ValidateAge*(n);  **if**(b==**true**)         {                System.***out***.println("TRUE");         }  **else**                System.***out***.println("FALSE");         s.close();  }  }  UserMainCode:  **public** **class** UserMainCode {  **public** **static** **boolean** ValidateAge(String n)  {  **boolean** b = **false**;  **if**(n.matches("[0-9]{2}"))         {                                                //Regular Expression – 2 (Age Validator) pg.No:150  **int** a=Integer.*parseInt*(n);  **if**(a>0&&a>=21&&a<=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  Main:  **import java.util.\*;**  **public class Main {**  **public static void main(String[]args){**  **Scanner s=new Scanner(System.in);**  **String s1=s.nextLine();**  **boolean b1=UserMainCode.validatePhone(s1);**  **if(b1==true)**  **{                                                          //phone validation** [**pg.no:151**](http://pg.no:151/)  **System.out.println("TRUE");**  **}**  **else**  **{**  **System.out.println("FALSE");**  **}**  **s.close();**  **}**  **}**  **UserMainCode:**  **public class UserMainCode {**  **public static boolean validatePhone(String s1)**  **{**  **boolean b=false;**  **if(s1.matches("[0]{1}[0]{1}[0-9]{8}") )**  **{**  **b=false;**  **}**  **//phone validation** [**pg.no:151**](http://pg.no:151/)  **else**  **{**  **b=true;**  **}**  **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  **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String[] args)  {         Scanner s = **new** Scanner(System.***in***);  String ip1=s.next();  **char** ip2='/';  String op[]=UserMainCode.*manipulateLiteral*(ip1,ip2);  **for**(**int** i=0;i<op.length;i++)  System.***out***.println(op[i]);  s.close();  }}  **import** java.util.ArrayList;  **import** java.util.StringTokenizer;  **public** **class** UserMainCode  {  **public** **static** String[] manipulateLiteral(String ip1, **char** ip2)  {  StringTokenizer t1 = **new** StringTokenizer(ip1,"/");  ArrayList<String> lst = **new** ArrayList<String>();  **while**(t1.hasMoreTokens())  {  StringBuffer sb = **new** StringBuffer();  sb.append(t1.nextToken().toLowerCase());  lst.add(sb.reverse().toString());  }  String[] op = **new** String[lst.size()];  **for**(**int** i = 0;i<lst.size();i++)  {  op[i] = lst.get(i);  }  **return** op;  }  }  **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.\*;  **public** **class** Main  {  **public** **static** **void** main(String[]args)              // Second set: 40.Vowel Count//         {  Scanner sc=**new** Scanner(System.***in***);  String s=sc.nextLine();  **int** max=UserMainCode.*tellVowelCount*(s);  System.***out***.println(max);  sc.close();  }  }  **public** **class** UserMainCode {  **public** **static** **int** tellVowelCount(String s)         {  **int** max=0;  **int** count=0;  **for**(**int** i=0;i<s.length();i++)         {  **char** c=s.charAt(i);  **if**(c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||c=='A'||c=='E'||c=='I'||         c=='O'||c=='U')         {         count++;         }         }  **if**(count>max)         {         max=count;         }  **return** max;         }  }    **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  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                Scanner s=**new** Scanner(System.***in***);  **int** n=s.nextInt();                String s1[]=**new** String[n];                String s2[]=**new** String[n];  **for**(**int** i=0;i<n;i++)                         //S.41.Playing with String - II//                {                       s1[i]=s.next();                }          s2=UserMainCode.*sortArray*(s1,n);  **for** (**int** i = 0; i < n; i++) {                       System.***out***.println(s2[i]);                }                s.close();         }}  **import** java.util.Arrays;  **public** **class** UserMainCode  {  **public** **static** String[] sortArray(String s1[],**int** n){                  String s2[]=**new** String[n];  **for** (**int** i = 0; i < n; i++)         {                s2[i]=s1[i].toLowerCase();         }         Arrays.*sort*(s2);  **return** s2;         }         }    **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  Main  **import** java.util.\*;  **public** **class** Main         {  **public** **static** **void** main(String[] args)         {  **int** n;         Scanner sin = **new** Scanner(System.***in***);         n = sin.nextInt();  **int**[] a1 = **new** **int**[n];  **for**(**int** i=0;i<n;i++)         {         a1[i] = sin.nextInt();         }         System.***out***.println(""+UserMainCode.*calculateMedian*(a1));         sin.close();         }         }  UserMainCode  **import** java.util.Arrays;  **public** **class** UserMainCode         {  **public** **static** **int** calculateMedian(**int**[] a)         {         Arrays.*sort*(a);  **int** length = a.length;  **int** result=0,mid=0,midNext=0;  **if**((length%2) != 0)         {         mid = (length/2)+1;         result = a[mid];         }  **else**         {         mid = length/2;         midNext = mid+1;  **float** add = a[mid-1]+a[midNext-1];  **float** div = add/2;         result = Math.*round*(div);         }  **return** result;         }    }  **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  Main  **import** java.util.\*;  **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();         }         System.***out***.println(UserMainCode.search*sequence*(a));         s.close();         }         }   UserMainCode  **public** **class** UserMainCode {  **public** **static** **boolean** searchsequence(**int**[] a)                {  **boolean** b = **false**;  **for**(**int** i = 0 ; i< a.length-3; i++)                {  **if**(a[i]==1 && a[i+1]==2 && a[i+2]==3)                b = **true**;                }  **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  Main  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                Scanner s=**new** Scanner(System.***in***);         String input=s.next();         System.***out***.println( UserMainCode.scanStarNeighbors (input));         s.close();         }           }         UserMainCode  **import** java.util.StringTokenizer;      **public** **class** UserMainCode {  **public** **static** **boolean** scanStarNeighbors(String input) {  **boolean** b=**false**;                StringTokenizer t=**new** StringTokenizer(input,"\*");                String s1=t.nextToken();                String s2=t.nextToken();                String s3=s1.substring(s1.length()-1);                String s4=s2.substring(0,1);  **if**(s3.equalsIgnoreCase(s4))                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  Main  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                Scanner s=**new** Scanner(System.***in***);         String s1=s.nextLine();         String s2=s.nextLine();  **int** v=UserMainCode.*countWords*(s1,s2);         System.***out***.println(v);         s.close();         }         }  UserMainCode  **import** java.util.StringTokenizer;  **public** **class** UserMainCode {  **public** **static** **int** countWords(String s1,String s2){                StringTokenizer t=**new** StringTokenizer(s1," ");  **int** c=0;  **while**(t.hasMoreTokens())         {         String s3=t.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    Main    **import** java.util.Scanner;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                Scanner s=**new** Scanner(System.***in***);         String s1=s.next();         String s2=s.next();  **int** v=UserMainCode.searchString(s1,s2);         System.***out***.println(v);         s.close();         }         }  UserMainCode  **public** **class** UserMainCode {  **public** **static** **int** searchString(String s1,String s2){  **int** c=0;  **int** t=s2.length();  **for**(**int** i=0;i<s1.length()-t+1;i++){  **if**(s2.equals(s1.substring(i,t+i))){                       c++;                }                }  **return** c;                }           }  **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    Main  **import** java.util.Scanner;  **public** **class** Main         {  **public** **static** **void** main(String args[])         {         String str=**new** String();         Scanner sc=**new** Scanner(System.***in***);         str=sc.nextLine();  **int** n=sc.nextInt();         String k=UserMainCode.*findFruitName*(str, n);         System.***out***.println(k);         sc.close();         }  }  UserMainCode  **import** java.util.StringTokenizer;  **public** **class** UserMainCode         {  **public** **static** String findFruitName(String m,**int** n)         {  **int** i=0;         String h=**null**;         StringTokenizer st=**new** StringTokenizer(m,",");  **int** max=st.countTokens();         String[] ss=**new** String[max];  **while**(st.hasMoreElements())         {         ss[i++]=st.nextToken();         }  **if**(n>max)         h=ss[i-1];  **else**         h=ss[n-1];  **return** h;         }         }  **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  Main  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args){                Scanner s=**new** Scanner(System.***in***);         String s1=s.nextLine();         System.***out***.println(UserMainCode.*changeCase*(s1));         s.close();         }    }  UserMainCode  **import** java.util.StringTokenizer;      **public** **class** UserMainCode {  **public** **static** String changeCase(String s1){                StringBuffer s5=**new** StringBuffer();                StringTokenizer t=**new** StringTokenizer(s1," ");  **while**(t.hasMoreTokens()){                String s2=t.nextToken();                String s3=s2.substring(0,1);                String s4=s2.substring(1, s2.length());                s5.append(s3.toUpperCase()).append(s4).append(" ");                }  **return** s5.toString();                }                }  **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    Main  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {                Scanner sc=**new** Scanner(System.***in***);           // **TODO** Auto-generated method stub         String s1=sc.nextLine();         System.Out.Println(UserMainCode.*compareLastWords*(s1));         sc.close();         }  }  UserMainCode  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.StringTokenizer;  **public** **class** UserMainCode {  **public** **static** **int** compareLastWords(String s1){         List<String> l=**new** ArrayList<String>();         StringTokenizer t=**new** StringTokenizer(s1," ");  **while**(t.hasMoreTokens())         {         String s2=t.nextToken();         l.add(s2);         }         String s3=l.get(0);         String s4=l.get(l.size()-1);  **if**(s3.equals(s4))         {  **int** n=s3.length();         System.***out***.println(n);         }  **else**         {  **int** n1=s3.length();  **int** n2=s4.length();  int n=n1+n2;         }  Return n;         }         }  **50.Perfect Number**  Write a program to that takes a positive integer and returns 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    import java.util.\*;  public class Main {  public static void main(String[] args){  Scanner s=new Scanner(System.in);  int n=s.nextInt();  boolean j=(UserMainCode.getPerfection(n));  if(j==true)  System.out.println("TRUE");  else  System.out.println("FALSE");  }  }  public class UserMainCode {  public static boolean getPerfection(int n){  boolean b=false;  int sum=0;  for(int i=1;i<n;i++){  int r=n%i;  if(r==0)  sum=sum+i;  }  b=(sum==n);  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    import java.util.\*;  public class Main {  public static void main(String[] args) {    Scanner s=new Scanner(System.in);  double d=s.nextDouble();  System.out.println(UserMainCode.findNoDigits(d));  }}      import java.util.StringTokenizer;      public class UserMainCode {  public static String findNoDigits(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](https://maps.google.com/?q=n2%21%3D1&entry=gmail&source=g))  if(s2.charAt(s2.length()-1)=='0')  n2=s2.length()-1;  String s3=String.valueOf(n1)+":"+String.valueOf(n2);  return s3;  }  }    **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    import java.util.Iterator;  import java.util.LinkedHashMap;  import java.util.Scanner;  public class Main {  public static void main(String[] args)  {  Scanner sc=new Scanner(System.in);  int k1=Integer.parseInt(sc.nextLine());  LinkedHashMap<String,String> hm=new LinkedHashMap<String,String>();  for(int i=0;i<k1;i++)  {  String k=sc.nextLine();  String s=sc.nextLine();  hm.put(k,s);  }  String n=sc.nextLine();  LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();  hm1=UserMainCode.obtainDesignation(hm,n);  Iterator<String> it=hm1.keySet().iterator();  while(it.hasNext())  {  String s2=it.next();  System.out.println(s2);  }  }  }      import java.util.HashMap;  import java.util.Iterator;  import java.util.LinkedHashMap;  import java.util.Map;  import java.util.Scanner;  public class UserMainCode  {  public static LinkedHashMap<String,String> obtainDesignation(LinkedHashMap<String,String> h1,String n)  {  int k=0;  LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();  Iterator<String>it=h1.keySet().iterator();  while(it.hasNext())  {  String s2=it.next();  String s3=h1.get(s2);  if(s3.equals(n))  hm1.put(s2,s3);  }  return hm1;  }}      **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.Iterator;  import java.util.LinkedHashMap;  import java.util.Scanner;  public class Main {  public static void main(String[] args)  {  Scanner sc=new Scanner(System.in);  int k1=Integer.parseInt(sc.nextLine());  LinkedHashMap<String,String> hm=new LinkedHashMap<String,String>();  for(int i=0;i<k1;i++)  {  String k=sc.nextLine();  String s=sc.nextLine();  hm.put(k,s);  }  String n=sc.nextLine();  LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();  hm1=UserMainCode.obtainDesignation(hm,n);  Iterator<String> it=hm1.keySet().iterator();  while(it.hasNext())  {  String s2=it.next();  System.out.println(s2);  }  }  }  import java.util.HashMap;  import java.util.Iterator;  import java.util.LinkedHashMap;  import java.util.Map;  import java.util.Scanner;  public class UserMainCode  {  public static LinkedHashMap<String,String> obtainDesignation(LinkedHashMap<String,String> h1,String n)  {  int k=0;  LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();  Iterator<String>it=h1.keySet().iterator();  while(it.hasNext())  {  String s2=it.next();  String s3=h1.get(s2);  if(s3.equals(n))  hm1.put(s2,s3);  }  return hm1;  }}  **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 Main {  public static void main(String[] args)  {  String str=new String();  Scanner sc=new Scanner(System.in);  str=sc.nextLine();  Boolean b=UserMainCode.ValidateDOB(str);  if(b==”true”)                  System.out.println("TRUE");  if(b==”false”)                  System.out.println("FALSE");  }  }    import java.text.SimpleDateFormat;  import java.util.Date;  public class UserMainCode {  public static Boolean ValidateDOB(String str){                  Boolean b=”false”;                  SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");                  sdf.setLenient(false);                  try                  {                  Date d1=sdf.parse(str);                  return b=”true”;                  }                  catch(Exception e)                  {                  return b=”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.println(UserMainCode.validateExp(s,s1));  }  }    import java.util.Calendar;  import java.util.Date;  public class UserMainCode {  public static boolean validateExp(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.\*;  **public** **class** Main  {  **public** **static** **void** main(String[] args)         {                Scanner s=**new** Scanner(System.***in***);                ArrayList<String> l=**new** ArrayList<String>();  **int** n=s.nextInt();  **for**(**int** i=0;i<n;i++)                {                       l.add(s.next());                }                String a[]=**new** String[n];                a=UserMainCode.*convertToStringArray*(l);  **for**(**int** j=0;j<n;j++)                {                       System.***out***.println(a[j]);                }         }  }    **import** java.util.ArrayList;  **import** java.util.Collections;    **class** UserMainCode  {  **public** **static** String[] convertToStringArray(ArrayList<String> l)         {                Collections.*sort*(l);                String [] a = l.toArray(**new** String[l.size()]);  **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  **Main Class**  **import java.util.\*;**  **public class Main**  **{**  **public static void main(String[] args)**  **{**  **Scanner s=new Scanner(System.*in*);**  **int n=s.nextInt();**  **String[] s1=new String[n];**  **for(int i=0;i<n;i++)**  **{**  **s1[i]=s.next();**  **}**  **HashMap<String, String> hm = new HashMap<String, String>();**  **hm = UserMainCode.*putvalues*(s1);**  **for(Map.Entry<String, String> ans: hm.entrySet())**  **{**  **System.*out*.println(ans.getKey()+":"+ans.getValue());**  **}**  **}}**  **User main code**  **import java.util.ArrayList;**  **import java.util.HashMap;**  **import java.util.Map;**  **public class UserMainCode{**  **public static HashMap<String, String> putvalues(String[] s1)**  **{**  **HashMap<String, String> hm = new HashMap<String, String>();**  **ArrayList<String> lst1 = new ArrayList<String>();**  **ArrayList<String> lst2 = new ArrayList<String>();**  **for(String s : s1)**  **lst1.add(s.toUpperCase().substring(0,3));**  **for(String s : s1)**  **lst2.add(s);**  **for(int i=0;i<s1.length;i++)**  **{**  **hm.put(lst1.get(i),lst2.get(i));**  **}**  **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    USERMAINCODE:    **import** java.util.ArrayList;  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** String[] fruitSelector(ArrayList<String> a1,ArrayList<String> a2)  {  ArrayList<String> a3=**new** ArrayList<String>();  **for**(**int** i=0;i<a1.size();i++)  {  String s1=a1.get(i);  **if**(s1.charAt(s1.length()-1)!='a'&&s1.charAt(s1.length()-1)!='e'&&s1.charAt(s1.length()-1)!='A'&&s1.charAt(s1.length()-1)!='E')  {  a3.add(s1);  }  }  ArrayList<String> a4=**new** ArrayList<String>();  **for**(**int** j=0;j<a2.size();j++)  {  String s2=a2.get(j);  **if**(s2.charAt(0)!='m'&&s2.charAt(0)!='a'&&s2.charAt(0)!='M'&&s2.charAt(0)!='A')  {  a4.add(s2);  }  }  a3.addAll(a4);  Collections.*sort*(a3);  String st[]=**new** String[a3.size()];  **for**(**int** k=0;k<a3.size();k++)  {  st[k]=a3.get(k);  }  **return** st;  }  }    MAIN:  **import** java.util.\*;  **import** java.util.ArrayList;  **public** **class** Main {  **public** **static** **void** main(String [] args)  {  Scanner s=**new** Scanner(System.***in***);  **int** m=s.nextInt();  ArrayList<String> aa1=**new** ArrayList<String>();  **for**(**int** i=0;i<m;i++)  {  aa1.add(s.next());  }  **int** n=s.nextInt();  ArrayList<String> aa2=**new** ArrayList<String>();  **for**(**int** j=0;j<n;j++)  {  aa2.add(s.next());  }  **int** k;  String st[]=UserMainCode.*fruitSelector*(aa1,aa2);  **for**( k=0;k<st.length;k++)  {  System.***out***.println(st[k]);  }  **if**(st.length==0)  System.***out***.println("No Fruit Found");  s.close();  }  }  **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  **MAIN:**  **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String[] args)  {  **int** n,m;  Scanner sin = **new** Scanner(System.***in***);  n = sin.nextInt();  ArrayList<Integer> a1 = **new** ArrayList<Integer>(n);  **for**(**int** i=0;i<n;i++)  {  **int** k = sin.nextInt();  a1.add(k);  }  m = sin.nextInt();  ArrayList<Integer> a2 = **new** ArrayList<Integer>(m);  **for**(**int** i=0;i<m;i++)  {  **int** k = sin.nextInt();  a2.add(k);  }  **int**[] result = UserMainCode.arrayListSubtractor(a1,a2);  Arrays.*sort*(result);  **for**(**int** i=0;i<result.length;i++)  System.***out***.println(result[i]);  }  }  **USERMAINCODE:**  **import** java.util.ArrayList;  **public** **class** UserMainCode  {  **public** **static** **int**[] arrayListSubtractor(ArrayList<Integer> arrlist1,ArrayList<Integer>  arrlist2)  {  **int** count=0,key;  **int** max = arrlist1.size();  **if**(arrlist1.size() < arrlist2.size())  max = arrlist2.size();  ArrayList<Integer> temp = **new** ArrayList<Integer>(max);  **for**(**int** i=0;i<arrlist1.size();i++)  {  key = (**int**)arrlist1.get(i);  **if**(arrlist2.indexOf(key) == -1)  {  ++count;  temp.add(key);  }  }  **for**(**int** i=0;i<arrlist2.size();i++)  {  key = (**int**)arrlist2.get(i);  **if**(arrlist1.indexOf(key) == -1)  {  **if**(!temp.contains(key))  {  ++count;  temp.add(key);  }  }  }  **int**[] result = **new** **int**[count];  **for**(**int** i=0;i<count;i++)  result[i] = (**int**)temp.get(i);  **return** result;  }  }  **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  **MAIN:**  **import** java.util.HashMap;  **import** java.util.Scanner;      **public** **class** Main {  **public** **static** **void** main(String[] args) {                       Scanner S=**new** Scanner(System.***in***);  **int** n=S.nextInt();                         HashMap<String, Float> m1=**new** HashMap<String, Float>();  **for**(**int** i=0;i<n;i++)                       {                             String name=S.next();  **float** price=S.nextFloat();                             m1.put(name,price);                       }  **int** m=S.nextInt();                       String s[]=**new** String[m];  **for**(**int** j=0;j<m;j++)                       {                             s[j]=S.next();                       }                       System.***out***.println(UserMainCode.*getTheTotalCostOfPheripherals*(m1,s));                }  }  **USERMAINCODE:**  **import** java.util.HashMap;  **import** java.util.Iterator;      **public** **class** UserMainCode {  **public** **static** **float** getTheTotalCostOfPheripherals(HashMap<String,Float> m1, String[] s) {                Float f=(**float**) 0;                Iterator<String> i=m1.keySet().iterator();  **while**(i.hasNext()){                String s1=i.next();                Float f1=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  **MAIN:**  **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** Main {  **public** **static** **void** main(String[] args) **throws** IOException, ParseException {                Scanner S=**new** Scanner(System.***in***);                String s1=S.next();                UserMainCode.*getLastDayOfMonth*(s1);                }         }  **USERMAINCODE:**  **import** java.text.ParseException;  **import** java.text.SimpleDateFormat;  **import** java.util.Calendar;  **import** java.util.Date;      **public** **class** UserMainCode {  **public** **static** **void** getLastDayOfMonth(String s1) **throws** ParseException{         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  **MAIN:**  **import** java.io.IOException;  **import** java.text.ParseException;  **import** java.text.SimpleDateFormat;  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) **throws** IOException, ParseException {         Scanner S=**new** Scanner(System.***in***);         String s1=S.next();         UserMainCode.*isLeapyear*(s1);  }  }  **USERMAINCODE:**  **import** java.text.ParseException;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** java.util.GregorianCalendar;  **import** java.util.StringTokenizer;      **public** **class** UserMainCode {  **public** **static** **void** isLeapyear(String s1) **throws** ParseException{         SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");         GregorianCalendar g=**new** GregorianCalendar();         StringTokenizer t=**new** StringTokenizer(s1,"/");         String s2=t.nextToken();         String s3=t.nextToken();         String s4=t.nextToken();  **int** n1=Integer.*parseInt*(s4);         Date d1=sdf.parse(s1);  **boolean** b=g.isLeapYear(n1);         System.***out***.println(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  **MAIN:**  **import** java.util.Scanner;  **public** **class** Main {  **public** **static** **void** main(String[] args) {         Scanner S=**new** Scanner(System.***in***);         String s1=S.nextLine();         System.***out***.println(UserMainCode.*getLargestSpan*(s1));  }  }  **USERMAINCODE:**  **import** java.util.StringTokenizer;      **public** **class** UserMainCode {  **public** **static** **int** getLargestSpan(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    **MAIN:**  **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***.print(UserMainCode.*getLargestSpan*(a,n));  }}    **USERMAINCODE:**  **public** **class** UserMainCode {  **public** **static** **int** getLargestSpan(**int**[] x,**int** n)  {  **int** gap=0,max=0;  **for**(**int** i=0;i<n;i++)  {  **for**(**int** j=i+1;j<n;j++)  {  **if**(x[i]==x[j])  {  gap=j;  }  }  **if**(gap-i>max)  max=gap-i;  }  **return** max+1;  }  }    **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    **MAIN:**  **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.sumElements(a));  }}    **USERMAINCODE:**  **import** java.util.Iterator;  **import** java.util.LinkedHashSet;  **public** **class** UserMainCode {  **public** **static** **int** sumElements(**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$1213  Sample Output 1:  valid  **Main:**  import java.util.Scanner;  public class Main {  public static void main(String[] args)  {  Scanner s=new Scanner(System.in);  String a=s.next();  System.out.println(UserMainCode.passwordValidation(a));  s.close();  }  }  **UserMainCode:**  public class UserMainCode  {  public static String passwordValidation(String a)  {  String k;  if(a.matches(".\*[0-9]{1,}.\*")&&a.matches(".\*[@#$]{1,}.\*")&&a.length()>=8&&a.matches(".\*[A-Z]{1,}.\*")&&a.matches(".\*[a-z].\*"))  {  k="validinput";  }  else  {  k="Invalidinput";  }  return k;  }  }  **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  **Main:**  import java.util.Iterator;  import java.util.LinkedHashMap;  import java.util.Scanner;  public class Main  {  public static void main(String[] args)  {  Scanner s=new Scanner(System.in);  int a=Integer.parseInt(s.nextLine());;  int[]k=new int[a];  for(int i=0;i<a;i++)  {  k[i]=s.nextInt();  }  LinkedHashMap<Integer,Integer>hm=new LinkedHashMap<Integer,Integer>();  hm=UserMainCode.getFactorial(k);  Iterator<Integer> it=hm.keySet().iterator();  for(int i=0;i<a;i++)  {  int n=it.next();  int fac=hm.get(n);  System.out.println(n+":"+fac);  s.close();  }  }  }  **UserMainCode;**  import java.util.LinkedHashMap;  public class UserMainCode  {  public static LinkedHashMap<Integer,Integer> getFactorial(int[] k)  {  LinkedHashMap<Integer,Integer> hm1=new LinkedHashMap<Integer,Integer>();  for(int i=0;i<k.length;i++)  {  int u=1;  for(int j=1;j<=k[i];j++)  {  u=u\*j;  }  hm1.put(k[i],u);  }  return hm1;  }  }  **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  **Main;**  import java.util.Scanner;  public class Main  {  public static void main(String[] args)  {  Scanner s=new Scanner(System.in);  String s1=s.next();  String s2=s.next();  System.out.println(UserMainCode.getCombo(s1,s2));  s.close();  }  }  **UserMainCode;**  public class UserMainCode  {  public static String getCombo(String s1,String s2)  {  StringBuffer sb=new StringBuffer();  int p=s1.length();  int q=s2.length();  if(p>q)  {  sb.append(s1).append(s2).append(s1);  }  else  {  sb.append(s2).append(s1).append(s2);  }  return sb.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  **Main:**  import java.util.Scanner;  public class Main {  public static void main(String[] args)  {  Scanner s=new Scanner(System.in);  String a=s.nextLine();  System.out.println(UserMainCode.getAge(a));  s.close();  }  }  **UserMainCode:**  import java.text.SimpleDateFormat;  import java.util.Date;  import java.util.Scanner;  public class UserMainCode  {  public static String getAge(String n)  {  Scanner s=new Scanner(System.in);  int year=0;  String s1=s.next();  SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");  try  {  Date d=sdf.parse(n);  Date d1=sdf.parse(s1);  int y=d.getYear();  int y1=d1.getYear();  int m=d.getMonth();  int m1=d1.getMonth();  int day=d.getDate();  int day1=d1.getDate();  year=y1-y;  if(m>m1)  year--;  else if(m==m1)  {if(day<day1)  year--;  }  }  catch(Exception e)  {  e.printStackTrace();  }  if(year>18)  return "eligible";  else  return "not-eligible";  }  }  **1. Unique Even Sum**  Write a program to read an array, eliminate 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  **Main:**  **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();  }  System.***out***.println(UserMainCode.*addUniqueEven*(a));  s.close();  }  }  **UserMainCode;**  **import** java.util.Iterator;  **import** java.util.LinkedHashSet;  **public** **class** UserMainCode  {  **public** **static** **int** addUniqueEven(**int**[] a)  {  **int** sum=0;  LinkedHashSet<Integer> hm=**new** LinkedHashSet<Integer>();  **for**(**int** i=0;i<a.length;i++)  {  hm.add(a[i]);  }  Iterator<Integer> im=hm.iterator();  **while**(im.hasNext())  {  **int** b=im.next();  **if**(b%2==0)  sum=sum+b;  }  **if**(sum>0)  {  **return** sum;  }  **else**  **return** -1;  }  }  **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  **Main;**  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.checkPalindrome(s));  sc.close();  }  }  **UserMainCode;**  import java.util.Iterator;  import java.util.LinkedHashSet;  public class UserMainCode  {  public static int checkPalindrome(String s)  {  StringBuffer sb=new StringBuffer(s);  int k=0;  LinkedHashSet<Character>l1=new LinkedHashSet<Character>();  String s2=sb.reverse().toString();  if(s2.equals(s))  {  String s3=s2.toLowerCase();  for(int i=0;i<s3.length();i++)  {  l1.add(s3.charAt(i));  }  Iterator<Character> it=l1.iterator();  while(it.hasNext())  {  char a=it.next();  if(a=='a'||a=='e'||a=='i'||a=='o'||a=='u')  k++;  }  }  if(k>=2)  return 1;  else  return -1;  }  }  **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  **Main;**  import java.util.Scanner;  public class Main  {  public static void main(String[] args)  {  Scanner sc=new Scanner(System.in);  String s=sc.next();  String s1=sc.next();  System.out.println(UserMainCode.replacePlus(s,s1));  sc.close();  }  }  **UserMainCode;**  public class UserMainCode  {  public static String replacePlus(String s,String s1)  {  {  String s2=s.toLowerCase();  String s3=s1.toLowerCase();  StringBuffer sb=new StringBuffer();  for(int i=0;i<s.length();i++)  {  char c=s2.charAt(i);  if(s3.indexOf(c)==-1)  sb.append("+");  else  sb.append(s.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  **Main;**  import java.util.Scanner;  public class Main  {  public static void main(String[] args)  {  Scanner s=new Scanner(System.in);  String s1=s.nextLine();  System.out.println(UserMainCode.getLargestWord(s1));  s.close();  }  }  **UserMainCode;**  import java.util.StringTokenizer;  public class UserMainCode  {  public static String getLargestWord(String s1)  {  int max=0;  String s2=new String();  StringTokenizer t=new StringTokenizer(s1," ");  {  while(t.hasMoreTokens()){  String s3=t.nextToken();  int n=s3.length();  if(n>max){  max=n;  s2=s3;}  }  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    **UserMainCode**    **import** java.util.StringTokenizer;  **public** **class** UserMainCode  {  **public** **static** **void** countNoOfWords(String s1, String s2) {  **int** count=0;  StringTokenizer st=**new** StringTokenizer(s2," ");  String s3=st.nextToken();  String s4=st.nextToken();  //System.out.println(s4);  StringTokenizer st1=**new** StringTokenizer(s1," ");  **while**(st1.hasMoreTokens())  {  String s5=st1.nextToken();  **if**(s4.equals(s5))  {  count++;  }  }  System.***out***.println(count);  }  }    **Main**    **import** java.util.\*;  **public** **class** Main  {  /\*\*  \* **@param** args  \*/  **public** **static** **void** main(String[] args)  {  Scanner s=**new** Scanner(System.***in***);  String s1=s.nextLine();  String s2=s.nextLine();  UserMainCode.*countNoOfWords*(s1,s2);  s.close();  }  }    **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    **UserMainCode**    **import** java.util.ArrayList;  **import** java.util.Iterator;  **public** **class** UserMainCode  {  **public** **static** ArrayList<Integer> generateOddEvenList (ArrayList<Integer>al1,ArrayList<Integer>al2)  {  ArrayList<Integer>al3=**new** ArrayList<Integer>();  **for**(**int** i=0;i<al1.size();i++)  {  **if**(i%2==0)  al3.add(al2.get(i));  **else**  al3.add(al1.get(i));  }  **return** al3;  }  }    **Main**    **import** java.util.ArrayList;  **import** java.util.Iterator;  **import** java.util.Scanner;  **public** **class** Main {  **public** **static** **void** main(String []args){  Scanner sc=**new** Scanner(System.***in***);  **int** s=Integer.*parseInt*(sc.nextLine());  ArrayList<Integer>al1=**new** ArrayList<Integer>();  ArrayList<Integer>al2=**new** ArrayList<Integer>();  **for**(**int** i=0;i<s;i++)  al1.add(sc.nextInt());  **for**(**int** i=0;i<s;i++)  al2.add(sc.nextInt());  ArrayList<Integer>al3=**new** ArrayList<Integer>();  al3=UserMainCode.*generateOddEvenList*(al1,al2);  Iterator<Integer> it=al3.iterator();  **while**(it.hasNext())  {  **int** n=it.next();  System.***out***.println(n);  sc.close();  }  }  }    **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    **UserMainCode**      **import** java.util.HashSet;  **import** java.util.Iterator;  **import** java.util.LinkedHashSet;  **import** java.util.StringTokenizer;  **public** **class** UserMainCode  {  **public** **static** **void** removeDuplicates(String s1) {  **char** a[]=s1.toCharArray();  StringBuffer sb=**new** StringBuffer();  LinkedHashSet<Character>hs=**new** LinkedHashSet<Character>();  **for**(**int** i=0;i<a.length;i++)  {  hs.add(a[i]);  }  Iterator<Character>itr=hs.iterator();  **while**(itr.hasNext())  {  **char** o=itr.next();  **if**(o!=' ');  {  sb.append(o);  }  }  System.***out***.println(sb);  }  }    **Main**    **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args)  {         Scanner s=**new** Scanner(System.***in***);         String s1=s.nextLine();         UserMainCode.*removeDuplicates*(s1);         s.close();  }  }    **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    **UserMainCode**    **import** java.util.HashMap;  **import** java.util.Scanner;  **import** java.util.HashMap;  **import** java.util.Iterator;  **public** **class** UserMainCode {  **public** **static** **int** getAverageOfOdd(HashMap<Integer,Integer>h1)  {  **int** av=0,c=0,s=0;  Iterator<Integer> it=h1.keySet().iterator();  **while**(it.hasNext())  {  **int** a=it.next();  **if**(a%2!=0)  {  **int** b=h1.get(a);  s=s+b;  c++;  }  }  av=s/c;  **return** av;  }}    **Main**  **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String args[])  {  Scanner sc=**new** Scanner(System.***in***);  **int** n=sc.nextInt();  HashMap<Integer,Integer> h1=**new** HashMap<Integer,Integer>();  **for**(**int** i=0;i<n;i++)  {  h1.put(sc.nextInt(),sc.nextInt());  }  System.***out***.println(UserMainCode.*getAverageOfOdd*(h1));  sc.close();  }  }    **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.  **SampleInput1:**  2  2  programmer  3000  8  manager  50000  **SampleOutput1:**  8  55000    **UserMainCode**  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.HashMap;  **public** **class** UserMainCode  {**public** **static** HashMap<Integer,Integer> display(HashMap<Integer,String>hm,HashMap<Integer,Integer>hm1)  {  HashMap<Integer,Integer>hm3=**new** HashMap<Integer,Integer>();  Iterator<Integer> it=hm.keySet().iterator();  **while**(it.hasNext())  {  **int** id=it.next();  String name=hm.get(id);  **if**(name.equals("manager"))  {**int** salary=hm1.get(id)+5000;  hm3.put(id,salary);  }}  **return** hm3;  }  }    **Main**  **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***);  **int** s=Integer.*parseInt*(sc.nextLine());  HashMap<Integer,String>hm=**new** HashMap<Integer,String>();  HashMap<Integer,Integer>hm1=**new** HashMap<Integer,Integer>();  **for**(**int** i=0;i<s;i++)  {  **int** id=Integer.*parseInt*(sc.nextLine());  hm.put(id, sc.nextLine());  hm1.put(id,Integer.*parseInt*(sc.nextLine()));  }  HashMap<Integer,Integer>hm2=**new** HashMap<Integer,Integer>();  hm2=UserMainCode.*display*(hm,hm1);  Iterator<Integer> it=hm2.keySet().iterator();  **while**(it.hasNext())  {  **int** n=it.next();  **int** fac=hm2.get(n);  System.***out***.println(n);  System.***out***.println(fac);  }  }  }  **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    **UserMainCode**    **import** java.util.StringTokenizer;  **public** **class** UserMainCode  {  **public** **static** **int** check(String s)  {  **int** count=0;  String fin="";  StringTokenizer st=**new** StringTokenizer(s);  String ini=st.nextToken();  **while**(st.hasMoreTokens())  { fin=st.nextToken();  }  **if**(ini.equals(fin))  count=ini.length();  **else**  count=ini.length()+fin.length();  **return** count;  }  }    **Main**    **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***);  String age=sc.nextLine();  System.***out***.println(UserMainCode.*check*(age));  sc.close();  }}    **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    **UserMainCode**    **public** **class** UserMainCode  {  **public** **static** String concatCharacter(String[] a)  {  StringBuffer sb=**new** StringBuffer();  **for**(**int** i=0;i<a.length;i++)  sb.append(a[i].charAt(a[i].length()-1));  **return** sb.toString();  }  }    **Main**    **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String []args){  Scanner sc=**new** Scanner(System.***in***);  **int** s=Integer.*parseInt*(sc.nextLine());  String []a=**new** String[s];  **for**(**int** i=0;i<s;i++)  {  a[i]=sc.nextLine();  }  System.***out***.println(UserMainCode.*concatCharacter*(a));  sc.close();  }  }    **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    **UserMainCode**    **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;  **import** java.util.Scanner;  **public** **class** UserMainCode   {  **public** **static** **void** getAnagram(String s1,String s2)  {  List<Character> l1=**new** ArrayList<Character>();  List<Character> l2=**new** ArrayList<Character>();  String s3=s1.replace(" ","");  String s4=s2.replace(" ","");  String s5=s3.toUpperCase();  String s6=s4.toUpperCase();  **for** (**int** i = 0; i < s5.length(); i++)  {  l1.add(s5.charAt(i));  }  **for** (**int** i = 0; i < s6.length(); i++)  {  l2.add(s6.charAt(i));  }  Collections.*sort*(l1);  Collections.*sort*(l2);  // System.out.println(l1);  // System.out.println(l2);  **if**(l1.equals(l2))  System.***out***.println("Anagram");  **else**  System.***out***.println("Not Anagram");  }  }      **Main**  **import** java.util.ArrayList;  **import** java.util.Collections;  **import** java.util.List;  **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();           UserMainCode.*getAnagram*(s1,s2);    }  }  **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  2nd arguement 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  **import** java.util.Scanner;  **public** **class** Main{  **public** **static** **void** main (String[] args)  {  // your code goes here  Scanner sc = **new** Scanner(System.***in***);  String input1=sc.next();  String input2=sc.next();  System.***out***.println(UserMainCode.*calculateMeterReading*(input1,input2));  sc.close();  }}    **public** **class** UserMainCode {  **public** **static** **int** calculateMeterReading(String input1, String input2)         {  **int** n1=Integer.*parseInt*(input1.substring(5,input1.length()));  **int** n2=Integer.*parseInt*(input2.substring(5,input2.length()));  **int** n=Math.*abs*((n2-n1)\*4);  **return** n;         }         }  **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]  **import** java.text.ParseException;  **import** java.util.LinkedHashMap;  **import** java.util.Scanner;  **public** **class** Main  {  **public** **static** **void** main(String args[]) **throws** ParseException{ Scanner sc=**new** Scanner(System.***in***); **int**  n=Integer.*parseInt*(sc.nextLine());  LinkedHashMap<String,String>a1=**new** LinkedHashMap<String,String>(); **for**(**int** i=0;i<n;i++) {  a1.put(sc.nextLine(),sc.nextLine());  }  System.***out***.println(UserMainCode.*retirementEmployeeList*(a1));  }  }  **import** java.text.\*;  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** ArrayList<String> retirementEmployeeList(LinkedHashMap<String,String>a1) **throws** ParseException         {         ArrayList<String>al=**new** ArrayList<String>();         Iterator <String>it=a1.keySet().iterator();  **while**(it.hasNext())         {String s=it.next();         String s1=a1.get(s);         SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");         sdf.setLenient(**false**);  **try**{         Date d=**new** Date();         Date d1=**new** Date();         String a=s1;         String b="01/01/2014";         d=sdf.parse(a);         d1=sdf.parse(b);  **long** t=d.getTime();  **long** t1=d1.getTime();  **long** t3=t1-t;  **long** l1=(24 \* 60 \* 60 \* 1000);  **long** l=l1\*365;  **long** res=t3/l;  **if**(res>=60)         {         al.add(s);         }         }  **catch** (Exception e) {         e.printStackTrace();         }         }         Collections.*sort*(al);  **return** al;         }         }      **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  **import** java.util.\*;  **public** **class** Main{  **public** **static** **void** main(String[] args)  {  Scanner sc=**new** Scanner(System.***in***);  **int** n=sc.nextInt();  **int** v=UserMainCode.**getKaprekarNumber**(n);  **if** (v==1)  System.***out***.println("Kaprekar Number");  **else**         System.***out***.println("Not a Kaprekar Number");  }}    **public** **class** UserMainCode {    **public** **static** **int** **getKaprekarNumber**(**int** a)                {  **int** count=0,j=0;  **int** a1=a;  **while**(a1!=0)                {                count=count+1;                a1=a1/10;                }  **int** square=a\*a;                String s=Integer.*toString*(square);                String s1=s.substring(0,count);                String s2=s.substring(count);  **int** x=Integer.*parseInt*(s1);  **int** y=Integer.*parseInt*(s2);  **int** result =x+y;  **if**(result==a){                j=1;                }  **else**                {                j=2;                }  **return** j;                }}    **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  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  Scanner sc=**new** Scanner(System.***in***);         String s1 =sc.nextLine();  UserMainCode.*storeMaxVowelWord*(s1);    }  }  **import** java.util.StringTokenizer;      **public** **class** UserMainCode {  **public** **static** **void** storeMaxVowelWord(String s1) {  **int** i = 0;                StringTokenizer st = **new** StringTokenizer(s1," ");  **int** len = 0;  **int** count = 0;  **int** count2 = 0;                String s6 = **null**;  **while** (st.hasMoreTokens()) {                String s5 = st.nextToken();                len = s5.length();                count=0;  **for** (i = 0; i < len; i++) {  **if** (s5.charAt(i) == 'a' || s5.charAt(i) == 'e'|| s5.charAt(i) == 'i' || s5.charAt(i) == 'o'|| s5.charAt(i) == 'u'                ||s5.charAt(i) == 'A' ||s5.charAt(i) == 'E' ||s5.charAt(i) == 'I' ||s5.charAt(i) == 'O' ||s5.charAt(i) == 'U')                count++;                }  **if** (count > count2)                {                count2 = count;                s6 = s5;                }                }                System.***out***.println(s6);                }                }    **17.Unique Characters REPEATED**  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  **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String[] args)  {         Scanner sc=**new** Scanner(System.***in***);  String s1=sc.next();  UserMainCode.*checkUnique*(s1);  }  }  **import** java.util.\*;  **public** **class** UserMainCode {    **public** **static** **void** checkUnique(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--;         }         }  **if**(count>0)         {         sb.deleteCharAt(i);         i--;         l--;         }         }  **if**(sb.length()==0)         {         System.***out***.println(-1);         }  **else**         System.***out***.println(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  **import** java.util.Scanner;  **public** **class** Main{  **public** **static** **void** main (String[] args)  {  // your code goes here  Scanner sc = **new** Scanner(System.***in***);  **int** n = sc.nextInt();  System.***out***.println(UserMainCode.**addPrimeIndex**(n));  }}  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** **int** **addPrimeIndex**(**int** n) {                       Scanner sc=**new** Scanner(System.***in***);  **int**[] a = **new** **int**[n];  **for**(**int** i=0;i<n;i++){                       a[i] = sc.nextInt();                       }  **int** sum=0;  **int** count=0;  **int** sum\_count=0;  **for**(**int** i=0;i<a.length;i++)                {                count=0;  **for**(**int** j=1;j<=i;j++)                {  **if**(i%j==0)                {                count++;                }                }  **if**(count==2)                {                sum=sum+a[i];                sum\_count++;                }                }  **int** avg=sum/sum\_count;  **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  **Main:**  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=Integer.parseInt(sc.nextLine());  ArrayList<Integer>a1=new ArrayList<Integer>();  ArrayList<Integer>a2=new ArrayList<Integer>();  for(int i=0;i<n;i++)  a1.add(Integer.parseInt(sc.nextLine()));  for(int i=0;i<n;i++)  a2.add(Integer.parseInt(sc.nextLine()));  char c=sc.nextLine().charAt(0);  System.out.println(UserMainCode.performSetOperations(a1,a2,c));  }  }  **UserMainCode:**                    import java.util.ArrayList;  import java.util.ArrayList;                  public class UserMainCode {                  public static ArrayList<Integer> performSetOperations(ArrayList<Integer>a1,ArrayList<Integer>a2,char c)                  {                  ArrayList<Integer>op1=new ArrayList<Integer>();                  int k=0;                  switch(c)                  {                  case '+':                  a1.removeAll(a2);                  a1.addAll(a2);                  op1=a1;                  break;                  case '\*':                  a1.retainAll(a2);                  op1=a1;                  break;                  case '-':                  for(int i=0;i<a1.size();i++)                  {                  k=0;                  for(int j=0;j<a2.size();j++)                  {                  if(a1.get(i)==a2.get(j))                  k=1;                  }                  if(k==0)                  op1.add(a1.get(i));                  }                  break;                  }                  return op1;                  }}    }  **return** tm;  }  }  **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  **MAIN:**  **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***.print(UserMainCode.*getMaxSpan*(a,n));  }}  **USERMAINCODE:**  **class** UserMainCode {  **public** **static** **int** getMaxSpan(**int**[] x,**int** n)  {  **int** gap=0,max=0;  **for**(**int** i=0;i<n;i++)  {  **for**(**int** j=i+1;j<n;j++)  {  **if**(x[i]==x[j])  gap=j;  }  **if**(gap-i>max)  max=gap-i;  }  **return** max+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 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  **USERMAINCODE:**  **import** java.util.ArrayList;  **import** java.util.StringTokenizer;  **public** **class** UserMainCode  {  **public** **static** String highestScorer(ArrayList<String>s1)  {  **int** max=0;  String s4=**null**;  **for**(**int** i=0;i<s1.size();i++)  {  String s2=s1.get(i);  StringTokenizer t=**new** StringTokenizer(s2,"-");  String s3=t.nextToken();  **int** n1=Integer.*parseInt*(t.nextToken());  **int** n2=Integer.*parseInt*(t.nextToken());  **int** n3=Integer.*parseInt*(t.nextToken());  **int** n=n1+n2+n3;  **if**(n>max)  {  max=n;  s4=s3;  }  }  **return** s4;  }  }  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Scanner s=**new** Scanner(System.***in***);  **int** n=s.nextInt();  ArrayList<String> s1=**new** ArrayList<String>();  **for**(**int** i=0;i<n;i++)  {  s1.add(s.next());  }  System.***out***.println(UserMainCode.*highestScorer*(s1));  s.close();  }  }  **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  **USERMAINCODE:**  **import** java.util.StringTokenizer;  **public** **class** UserMainCode {  **public** **static** String getWordWithMaximumVowels(String s1)  {  **int** i;  StringTokenizer t=**new** StringTokenizer(s1," ");  **int** count=0,max=0;  String s2=**null**;  **while**(t.hasMoreTokens())  {  String s3=t.nextToken();  count=0;  **for**(i=0;i<s3.length();i++)  {  **if**(s3.charAt(i)=='a'||s3.charAt(i)=='e'||s3.charAt(i)=='i'||s3.charAt(i)=='o'||s3.charAt(i)=='u'  ||s3.charAt(i)=='A'||s3.charAt(i)=='E'||s3.charAt(i)=='I'||s3.charAt(i)=='O'||s3.charAt(i)=='U')  count++;    }  **if**(count>max)  {  max=count;  s2=s3;  }    }  **return** s2;  }  }  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Scanner s=**new** Scanner(System.***in***);  String s1=s.nextLine();  System.***out***.println(UserMainCode.*getWordWithMaximumVowels*(s1));  s.close();  }  }  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  **USERMAINCODE:**  **public** **class** UserMainCode {  **public** **static** **int** getOrderVowels(String s1)  {    String s2="aeiou";  StringBuffer sb=**new** StringBuffer();  **for**(**int** i=0;i<s1.length();i++)  {  **for**(**int** j=0;j<s2.length();j++)  {  **if**(s1.charAt(i)==s2.charAt(j))  {  sb.append(s1.charAt(i));  }  }  }  **if**(sb.toString().equals(s2))  {  **return** 1;  }  **return** -1;  }  }  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Scanner s=**new** Scanner(System.***in***);  String s1=s.next();  **int** b=UserMainCode.*getOrderVowels*(s1);  **if**(b==1)  {  System.***out***.println("Valid");  }  **else**  System.***out***.println("Invalid");  s.close();  }  }  **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  **USERMAINCODE:**  **import** java.util.\*;  **public** **class** UserMainCode {  **public** **static** String swapPairs(String s1)  {  **int** i;  StringBuffer sb=**new** StringBuffer();  **for**(i=0;i<s1.length()-1;i=i+2)  {  **if**(i%2==0)  {  **char** a=s1.charAt(i);  **char** b=s1.charAt(i+1);  sb.append(b).append(a);  }  **else**  **for**(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(s1.length()-1));  }    }  **return** sb.toString();  }  }  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Scanner s=**new** Scanner(System.***in***);  String s1=s.nextLine();  System.***out***.println(UserMainCode.*swapPairs*(s1));  s.close();  }  }  **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  **USERMAINCODE:**  **public** **class** UserMainCode {  **public** **static** **int** getDigits(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;  }  }  **return** sum;  }  }  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  Scanner s=**new** Scanner(System.***in***);  String s1=s.next();  System.***out***.println(UserMainCode.*getDigits*(s1));  s.close();  }  }  **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  **USERMAINCODE:**  **public** **class** UserMainCode {  **public** **static** **boolean** validatePassword(String s1)  {  **boolean** b=**false**;  **if**(s1.length()>=8)  b=**true**;  **if**(b=**true**)  {  **if**(s1.matches(".\*[0-9]{1,}.\*")&&s1.matches(".\*[a-zA-Z]{1,}.\*")&&s1.matches(".\*[@#%]{1,}.\*"))  {  b=**true**;  }  **else**  b=**false**;  }  **return** b;  }  }  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String [] args)  {  Scanner s=**new** Scanner(System.***in***);  String s1=s.nextLine();  **boolean** b=(UserMainCode.*validatePassword*(s1));  **if**(b==**true**)  {   System.***out***.println("Valid");  }  **else**  System.***out***.println("Invalid");  s.close();  }  }  **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  **USERMAINCODE:**  import java.text.ParseException; import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date; import java.util.Iterator; import java.util.LinkedHashMap; import java.util.TreeMap; class UserMainCode { public static TreeMap<Integer,Integer> calculateRevisedSalary(LinkedHashMap<Integer,String>a1,LinkedHashMap<Integer,Integer>a2) throws ParseException {TreeMap<Integer,Integer>ans=new TreeMap<Integer,Integer>(); ArrayList<String>al=new ArrayList<String>(); Iterator <Integer>it=a1.keySet().iterator(); while(it.hasNext()) {int s=it.next(); String s1=a1.get(s); SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy"); sdf.setLenient(false); try{ Date d=new Date(); Date d1=new Date(); String a=s1; String b="01-09-2014"; d=sdf.parse(a); d1=sdf.parse(b); long t=d.getTime(); long t1=d1.getTime(); long t3=t1-t; long l1=(24 \* 60 \* 60 \* 1000); long l=l1\*365; long res=t3/l; //System.out.println("Result="+res); if(res>=25 && res<=30) { float bonus=(float)((0.2\*a2.get(s))+a2.get(s)); int r=(int)bonus; ans.put(s,r ); }else if(res>30 && res<=60) { float bonus=(float)((0.3\*a2.get(s))+a2.get(s)); int r=(int)bonus; ans.put(s,r ); } else if(a2.get(s)<5000) { ans.put(s, -100); } else if(res<25 ||res>60) { ans.put(s, -200); } } catch (Exception e) { e.printStackTrace(); } } return ans; } }  **MAIN:**  import java.text.ParseException; import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date; import java.util.Iterator; import java.util.LinkedHashMap; import java.util.Scanner; import java.util.TreeMap; public class Main { public static void main(String args[]) throws ParseException{ Scanner sc=new Scanner(System.in); int n=sc.nextInt(); LinkedHashMap<Integer,String>a1=new LinkedHashMap<Integer,String>(); LinkedHashMap<Integer,Integer>a2=new LinkedHashMap<Integer,Integer>(); TreeMap<Integer,Integer>ans=new TreeMap<Integer, Integer>(); for(int i=0;i<n;i++) {int id=sc.nextInt(); a1.put(id,sc.next()); int salary=sc.nextInt(); a2.put(id,salary); } ans=UserMainCode.calculateRevisedSalary(a1,a2); Iterator <Integer>it=ans.keySet().iterator(); while(it.hasNext()) { int a=it.next(); int b=ans.get(a); System.out.println(a); System.out.println(b); } } }  **28. Grade Calculator REFER 53 FROM LEVEL2**  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  **USERMAINCODE:**  import java.util.Iterator; import java.util.HashMap; import java.util.TreeMap; public class UserMainCode { public static TreeMap<Integer,String>calculateGrade(HashMap<Integer,Integer>hm) { TreeMap<Integer,String>tm=new TreeMap<Integer,String>(); Iterator<Integer> it=hm.keySet().iterator(); while(it.hasNext()) { int id=it.next(); int mark=hm.get(id); if(mark>=80) tm.put(id,"GOLD"); else if(mark<80 && mark>=60) tm.put(id,"SILVER"); else if(mark<60 && mark>=45) tm.put(id,"BRONZE"); else tm.put(id,"FAIL"); } return tm; }}  **MAIN:**  import java.util.HashMap; import java.util.Iterator; import java.util.HashMap; import java.util.TreeMap; import java.util.Scanner; public class Main { public static void main(String []args){ Scanner sc=new Scanner(System.in); int s=sc.nextInt(); HashMap<Integer,Integer>hm=new HashMap<Integer,Integer>(); for(int i=0;i<s;i++) { hm.put(sc.nextInt(),sc.nextInt()); } TreeMap<Integer,String>tm=new TreeMap<Integer,String>(); tm=UserMainCode.calculateGrade(hm); Iterator<Integer> it=tm.keySet().iterator(); for(int i=0;i<s;i++) { int n=it.next(); String fac=tm.get(n); System.out.println(n); System.out.println(fac); } } }  29.Digits - II  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  **MAIN:**  import java.util.Scanner;  public class Main {  public static void main(String[] args)                  {                                  Scanner s=new Scanner(System.in);                                  int a=s.nextInt();                                  int sum=UserMainCode.getDigitSum(a);                                  System.out.println(sum);                  }  }  **USERMAINCODE:**  public class UserMainCode  {  public static int getDigitSum(int n)                  {                                  int sum = 0 ;                                  int n1=n;                                  while(n>10)                                  {                                                  int a = 0 ; sum = 0;                                                  while(n!=0)                                                  {                                                                  a = n%10;                                                                  sum+=a;                                                                  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  **MAIN:**  **import** java.util.\*;  **public** **class** Main {      **public** **static** **void** main(String[] args){    Scanner s=**new** Scanner(System.***in***);    String s1=s.nextLine();    String s2=s.nextLine();    System.***out***.println(UserMainCode.*checkAnagram*(s1,s2));    }    }  **USERMAINCODE:**  **import** java.util.\*;  **import** java.text.\*;  **public** **class** UserMainCode {  **public** **static** **boolean** checkAnagram(String s1,String s2)   {  **boolean** b=**false**;      String aj1 =s1.toLowerCase();                 //ANAGRAMS    String aj2=s2.toLowerCase();      ArrayList<Character> a1 = **new** ArrayList<Character>();    ArrayList<Character> a2  = **new** ArrayList<Character>();  **for**(**int** i=0;i<aj1.length();i++)    {  **char** c=aj1.charAt(i);  **if**(c!=' ')     {      a1.add(c);     }    }  **for**(**int** j=0;j<aj2.length();j++)    {  **char** c=aj2.charAt(j);  **if**(c!=' ')     {      a2.add(c);     }    }  **if**(a1.size()==a2.size())    {  **if**(a1.containsAll(a2))     {      b= **true**;     }    }    **return** b;   }    }      31.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  **MAIN:**  import java.util.Scanner;  public class Main {                  public static void main(String[] args)                  {                                  Scanner sc=new Scanner(System.in);                                  int size=sc.nextInt();                                  int[]m=new int[size];                                  int[]n=new int[size];                                  for(int i=0;i<size;i++)                                  {                                  n[i]=sc.nextInt();                                  }                                  m=UserMainCode.shiftLeft(n);                                  for(int i=0;i<size;i++)                                  {                                                  System.out.println(m[i]);                                  }                  }  }    **USERMAINCODE:**  **public** **class** UserMainCode {  **public** **static** **int**[] shiftLeft(**int** n[])         {  **int** j=0;  **int**[]m=**new** **int**[n.length];  **for**(**int** i=0;i<n.length;i++)                {  **if**(n[i]!=5)                       {                             m[j]=n[i];                             j++;                       }                }  **return** m;         }  }     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  **MAIN:**  import java.util.Scanner;  public class Main {  public static void main(String[] args)                  {                                  Scanner s=new Scanner(System.in);                                  int n=s.nextInt();                                  String a[]=new String[n];                                  for(int i=0;i<n;i++)                                  {                                                  a[i]=s.next();                                  }                                  System.out.println(UserMainCode.sumOfDigits(a));                  }  }  **USERMAINCODE:**  public class UserMainCode {                  public static int sumOfDigits(String[] s1)                  {                                  int sum = 0;                                  for(int i=0;i<s1.length;i++)                                  {                                                  String s = s1[i];                                                  for(int j = 0;j<s.length();j++)                                                  {                                                                  Character c = s.charAt(j);                                                                  if(Character.isDigit(c))                                                                  {                                                                                  sum+=Integer.parseInt(s.valueOf(c));                                                                  }                                                  }                                  }                                  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  **MAIN:**  import java.util.Scanner;  public class Main {                  public static void main(String[] args)                  {                                  Scanner sc = new Scanner(System.in);                                  int n=Integer.parseInt(sc.nextLine());                                  String s[]=new String[n];                                  for(int i=0;i<n;i++)                                  s[i]=sc.nextLine();                                  System.out.println(UserMainCode.findPrefix(s));                  }  }  **USERMAINCODE:**  import java.util.ArrayList;  import java.util.Iterator;  import java.util.LinkedHashSet;  public class UserMainCode  {                  public static int findPrefix (String s[]) {                                  LinkedHashSet<String>l1=new LinkedHashSet<String>();                                  ArrayList<String>a1=new ArrayList<String>();                                  int c=0;                                  for(int i=0;i<s.length;i++)                                  l1.add(s[i]);                                  Iterator<String> it=l1.iterator();                                  while(it.hasNext())                                  {                                  a1.add(it.next());                                  }                                  for(int i=0;i<a1.size();i++)                                  {                                  String s2=a1.get(i);                                  for(int j=0;j<a1.size();j++)                                  {                                  String s3=a1.get(j);                                  if(i!=j&&s3.length()>s2.length())                                  {                                  String s4=s3.substring(0,s2.length());                                  if(s2.equals(s4))                                  c++;                                  }                                  }                                  }                                  return c;                                  }                  }   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  **MAIN:**  **import** java.util.Scanner;    **public** **class** Main {    **public** **static** **void** main(String[] args) {                Scanner sc = **new** Scanner(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(UserMainCode.*countCommonStrings*(s1,s2));                }                }    **USERMAINCODE:**  **import** java.util.ArrayList;      **public** **class** UserMainCode {  **public** **static** **int** countCommonStrings(String[] s1,String[] s2)         {  **int** count=0;                ArrayList<String> al = **new** ArrayList<String>();  **for** (**int** i = 0; i < s1.length; i++) {  **for** (**int** j = 0; j < s2.length; j++) {  **if**(s1[i].equals(s2[j])){  **if**(!al.contains(s1[i])){                count++;                al.add(s1[i]);                }                }                }                }  **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:  7  **MAIN:**  import java.util.Scanner;  public class Main {           public static void main(String[] args)         {                Scanner s=new Scanner(System.in);                int n=s.nextInt();                System.out.println(UserMainCode.getFibonacciSum(n));                }           }  **USERMAINCODE:**  import java.util.ArrayList;                  import java.util.Scanner;                  public class UserMainCode {                  public static int 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.E-Mail 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  **MAIN:**  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {         Scanner s=**new** Scanner(System.***in***);         String ip;         ip=s.next();  **boolean** b=UserMainCode.*ValidateEmail*(ip);  **if**(b==**true**)                System.***out***.println("TRUE");  **else**                System.***out***.println("FALSE");  }}  **USERMAINCODE:**  **import** java.util.StringTokenizer;  **public** **class** UserMainCode {  **public** **static** **boolean** ValidateEmail(String ip) {  **int** i=0;  **boolean** b=**false**;         StringTokenizer t=**new** StringTokenizer(ip,"@");         String s1=t.nextToken();         String s2=t.nextToken();         StringTokenizer t1=**new** StringTokenizer(s2,".");         String s3=t1.nextToken();         String s4=t1.nextToken();  **if**(ip.contains("@") && ip.contains("."))         i++;  **if**(i==1)  **if**(s3.length()==5)  **if**(s1.length()>=3)  **if**(s4.equals("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    **MAIN:**  **import** java.util.\*;  **public** **class** Main  {  **public** **static** **void** main(String[] args)  {  **int** n,m;  Scanner sin = **new** Scanner(System.***in***);  n = sin.nextInt();  **int**[] a1 = **new** **int**[n];  **for**(**int** i=0;i<n;i++)  {  a1[i] = sin.nextInt();  }  m = sin.nextInt();  **int**[] a2 = **new** **int**[m];  **for**(**int** i=0;i<m;i++)  {  a2[i] = sin.nextInt();  }  **int**[] result = UserMainCode.*getSymmetricDifference* (a1,a2);  **for**(**int** i=0;i<result.length;i++)  System.***out***.println(result[i]);  }  }  **USERMAINCODE:**  **import** java.util.\*;  **public** **class** UserMainCode  {  **public** **static** **int**[] getSymmetricDifference (**int**[] a1,**int**[] a2)  {  //int[] a1 = new int[]{11,5,14,26,3};  //int[] a2 = new int[]{5,3,1};  **int**[] union,inter,result;  **int** count=0;  **int** max = a1.length+a2.length;  ArrayList<Integer> temp = **new** ArrayList<Integer>(max);  /\*union\*/  **for**(**int** i=0;i<a1.length;i++)  {  **if**(!temp.contains(a1[i]))  {  ++count;  temp.add(a1[i]);  }  }  **for**(**int** i=0;i<a2.length;i++)  {  **if**(!temp.contains(a2[i]))  {  ++count;  temp.add(a2[i]);  }  }  union = **new** **int**[count];  **for**(**int** i=0;i<count;i++)  {  union[i] = (**int**)temp.get(i);  }  Arrays.*sort*(union);  /\*intersection\*/  temp = **new** ArrayList<Integer>(max);  count =0;  Arrays.*sort*(a2);  **for**(**int** i=0;i<a1.length;i++)  {  **if**(Arrays.*binarySearch*(a2,a1[i]) >= 0)  {  ++count;  temp.add(a1[i]);  }  }  inter = **new** **int**[count];  **for**(**int** i=0;i<count;i++)  {  inter[i] = (**int**)temp.get(i);  }  Arrays.*sort*(inter);  /\*difference \*/  temp = **new** ArrayList<Integer>(max);  count =0;  Arrays.*sort*(inter);  **for**(**int** i=0;i<union.length;i++)  {  **if**(Arrays.*binarySearch*(inter,union[i]) < 0)  {  ++count;  temp.add(union[i]);  }  }  result = **new** **int**[count];  **for**(**int** i=0;i<count;i++)  {  result[i] = (**int**)temp.get(i);  }  Arrays.*sort*(result);  //System.out.println("resultant array : \n "+Arrays.toString(result));  **return** result;  }  }      **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    **MAIN:**    **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.*getDayofWeek*(s1));  }  }      **USERMAINCODE:**    **import** java.text.SimpleDateFormat;  **import** java.text.ParseException;  **import** java.util.Date;  **public** **class** UserMainCode {  **public** **static** String getDayofWeek(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.toLowerCase();  }  }      **39.Add Time**    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    **MAIN:**    **import** java.util.\*;  **import** java.io.IOException;  **import** java.text.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) **throws** IOException, ParseException {         Scanner s = **new** Scanner(System.***in***);  String s1=s.next();  String s2=s.next();  System.***out***.println(UserMainCode.*addTime*(s1,s2));  }    }    **USERMAINCODE:**    **import** java.util.\*;  **import** java.io.IOException;  **import** java.text.\*;      **public** **class** UserMainCode {  **public** **static** String addTime(String s1,String s2) **throws** IOException, ParseException{                SimpleDateFormat sdf=**new** SimpleDateFormat("HH:mm:ss");                sdf.setTimeZone(TimeZone.*getTimeZone*("UTC"));                sdf.setTimeZone(TimeZone.*getTimeZone*("s1"));                sdf.setTimeZone(TimeZone.*getTimeZone*("s2"));                Date d1=sdf.parse(s1);                Date d2=sdf.parse(s2);  **long** add=d1.getTime()+d2.getTime();                String s=sdf.format(add);                Calendar cal=Calendar.*getInstance*();                cal.setTime(sdf.parse(s));  **int** FindDay=cal.get(Calendar.***DAY\_OF\_MONTH***);  **if**(FindDay>1)                FindDay=FindDay-1;                String op=FindDay+":"+s;  **return** op;                }  }      **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    **MAIN:**    **import** java.util.\*;    **public** **class** Main {  **public** **static** **void** main(String[] args) {         Scanner s = **new** Scanner(System.***in***);  String ip=s.next();  System.***out***.println(UserMainCode.*ISBNnumber*(ip));    }  }    **USERMAINCODE:**    **import** java.util.\*;  **import** java.text.\*;      **public** **class** UserMainCode {  **public** **static** String ISBNnumber(String ip) {                String b="FALSE";  **int** sum=0;  **for**(**int** i=0,j=ip.length();i<ip.length();i++,j--){                String s=String.*valueOf*(ip.charAt(i));  **int** n=Integer.*parseInt*(s);                sum+=(n\*j); }                //System.out.println(sum);  **if**(sum%11==0)                b="TRUE";  **return** b;                }                }      **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    **MAIN:**    **import** java.text.ParseException;  **import** java.text.SimpleDateFormat;  **import** java.util.\*;  **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(UserMainCode.*findOldDate*(s1,s2));  }  }    **USERMAINCODE:**    **import** java.text.ParseException;  **import** java.text.SimpleDateFormat;  **import** java.util.Calendar;  **import** java.util.Date;      **public** **class** UserMainCode {  **public** **static** String findOldDate(String s1,String s2) **throws** ParseException {         SimpleDateFormat sdf=**new** SimpleDateFormat("dd/MM/yyyy");         SimpleDateFormat sdf1=**new** SimpleDateFormat("MM-dd-yyyy");         Date d1=sdf.parse(s1);         Date d2=sdf.parse(s2);         Calendar cal=Calendar.*getInstance*();         cal.setTime(d1);  **long** y=cal.getTimeInMillis();         cal.setTime(d2);  **long** y1=cal.getTimeInMillis();         String s3=sdf1.format(d1);         String s4=sdf1.format(d2);  **if**(y<y1)  **return** s3;  **else**  **return** s4;         }  }              **42.Interest Calculation**    Write a program to calculate amount of the acccount holders based on the below mentioned prototype:  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    **MAIN**    **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** 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.*calculateInterestRate*(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);  }  }  }      **USERMAINCODE**    **import** java.text.DecimalFormat;  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.HashMap;  **import** java.util.TreeMap;  **public** **class** UserMainCode  {  **public** **static** TreeMap<String,Integer> calculateInterestRate (HashMap<String,String>hm,HashMap<String,Integer>hm1)  {  **int** year=0,amount=0;  **double** dis=0;  String now="01/01/2015";  TreeMap<String,Integer>tm=**new** TreeMap<String,Integer>();  Iterator<String> it=hm.keySet().iterator();  **while**(it.hasNext())  {  String id=it.next();  String dor=hm.get(id);  amount=hm1.get(id);  SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");  SimpleDateFormat sdf1=**new** SimpleDateFormat("dd/MM/yyyy");  **try**  {  Date d=sdf.parse(dor);  Date d1=sdf1.parse(now);  sdf.setLenient(**false**);  **int** y=d.~~getYear~~();  **int** y1=d1.~~getYear~~();  **int** m=d.~~getMonth~~();  **int** m1=d1.~~getMonth~~();  **int** day=d.~~getDay~~();  **int** day1=d1.~~getDay~~();  year=y1-y;  **if**(m>m1)  year--;  **else** **if**(m==m1)  {**if**(day<day1)  year--;  }  **if**(year>=60)  dis=0.1\*amount+amount;  **else** **if**(year<60 && year>=30 )  dis=0.07\*amount+amount;  **else**  dis=0.04\*amount+amount;  tm.put(id,(**int**)dis);  }  **catch**(Exception e)  {  e.printStackTrace();  }  }  **return** tm;  }  }          **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  **MAIN:**  **import** java.util.HashMap;  **import** java.util.Iterator;  **import** java.util.TreeMap;  **import** java.util.Scanner;  **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.*calculateDiscount*(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);  }  }  }  **USERMAINCODE**  **import** java.text.SimpleDateFormat;  **import** java.util.Date;  **import** java.util.HashMap;  **import** java.util.\*;  **public** **class** UserMainCode  {  **public** **static** TreeMap<String,Integer> calculateDiscount (HashMap<String,String>hm,HashMap<String,Integer>hm1)  {  **int** amount=0;  **double** dis=0;  String now="01/01/2015";  TreeMap<String,Integer>tm=**new** TreeMap<String,Integer>();  Iterator<String> it=hm.keySet().iterator();  **while**(it.hasNext())  {  String id=it.next();  String dor=hm.get(id);  amount=hm1.get(id);  SimpleDateFormat sdf=**new** SimpleDateFormat("dd-MM-yyyy");  sdf.setLenient(**false**);  **try**{  Date d=**new** Date();  Date d1=**new** Date();  String a=dor;  String b="01-01-2014";  d=sdf.parse(a);  d1=sdf.parse(b);  **long** t=d.getTime();  **long** t1=d1.getTime();  **long** t3=t1-t;  **long** l1=(24 \* 60 \* 60 \* 1000);  **long** l=l1\*365;  **long** year1=t3/l;  //System.out.println("Result="+year1);  **if**(year1>=5 && amount>=20000)  dis=0.2\*amount;  **else** **if**(year1<5 && amount>=20000)  dis=0.1\*amount;  **else** **if**(year1>=5 && amount<20000)  dis=0.15\*amount;  **else**  dis=0.05\*amount;  tm.put(id,(**int**)dis);  }  **catch**(Exception e)  {  e.printStackTrace();  }  }  **return** tm;  }  } |