**1.      Remove 10's**

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

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

Create a Class Main which would be used to read the number of elements and the input array, and call the static method present in UserMainCode.

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

**Sample Input :**

5

1

10

20

10

2

**Sample Output :**

1

20

2

o

o

import java.util.Scanner;

public class Pro1 {

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();

}

Integer[] s=UserMainCode.getDigitSum(n,a);

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

{

System.out.println(s[i]);

}

}

}

import java.util.ArrayList;

import java.util.Iterator;

public class UserMainCode {

public static Integer[] getDigitSum(int n,int[] a) {

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

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

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

{

al.add(a[i]);

}

Iterator<Integer> itr=al.iterator();

while(itr.hasNext())

{

int x=itr.next();

if(x!=10)

{

op.add(x);

}

}

if(op.size()<n)

{

n=n-op.size();

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

{

op.add(0);

}

}

Integer m[]=new Integer[op.size()];

op.toArray(m);

return m;

}

}

**2.      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

Scanner sc=new Scanner(System.in);

int a=sc.nextInt();

int b=sc.nextInt();

int c=sc.nextInt();

int sum=0;

if(a==13)

{

sum=c;

}

else if(b==13)

{

sum=a;

}

else if(c==13)

{

sum=a+b;

}

else

{

sum=a+b+c;

}

System.out.println(sum);

**3.      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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String s=sc.next();

StringBuffer sb=new StringBuffer();

if(s.charAt(0)=='j')

{

if(s.charAt(1)=='b')

{

sb.append(s);

}

else

{

sb.append(s.charAt(0)).append(s.substring(2));

}

}

else if(s.charAt(1)=='b')

{

sb.append(s.charAt(1)).append(s.substring(2));

}

else

{

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

}

System.out.println(sb);

}

}

4. **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

**import** java.io.\*;

**import** java.util.\*;

**public** **class** Main {

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

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

String s1=sc.next();

**if**(s1.matches("[#]{1}[A-F0-9]{6}"))

{

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

}

**else**

{

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

}

}

}

5.       **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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

int rem,sum=0,dsum=0,rem1;

Scanner sc=new Scanner(System.in);

int digit=sc.nextInt();

while(digit!=0)

{

rem=digit%10;

sum=sum+rem;

digit/=10;

}

if(sum<9)

{

System.out.println(sum);

}

else

{

while(sum!=0)

{

rem1=sum%10;

dsum+=rem1;

sum/=10;

}

System.out.println(dsum);

}

}

}

6. **Add and Reverse**

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

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

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

**Example:**

Input Array = {10,15,20,25,30,100}

Number = 15

sum = 20 + 25 + 30 + 100 = 175

output = 571

**Input and Output Format:**

The first line of the input consists of an integer that corresponds to the number of elements in the array.

The next n lines of the input consists of integers that correspond to the elements in the array.

The last line of the input consists of an integer that corresponds to the number.

Output consists of a single integer.

**Sample Input**

6

10

15

20

25

30

100

15

**Sample Output**

571

**import** java.io.\*;

**import** java.util.\*;

**public** **class** Main {

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

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

**int** s=sc.nextInt();

**int** rev=0,sum=0;

**int** a[]=**new** **int**[s];

**for**(**int** i=0;i<s;i++)

{

a[i]=sc.nextInt();

}

**int** ss=a.length;

**int** n=sc.nextInt();

**for**(**int** i=ss-1;i>0;i--)

{

**if**(n==a[i])

{

**break**;

}

**else**

{

sum+=a[i];

}

}

System.*out*.println(sum);

**int** temp;

temp=sum;

**while**(temp!=0)

{

**int** rem=temp%10;

rev=rev\*10+rem;

temp/=10;

}

System.*out*.println(rev);

}

}

7.       **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

**import** java.io.\*;

**import** java.util.\*;

**public** **class** Main {

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

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

String s1=sc.next();

String s2=sc.next();

String s3;

StringBuffer sb2=**new** StringBuffer(s2);

sb2.reverse();

s2=sb2.toString();

**int** a=sc.nextInt();

s1=String.*valueOf*(s1.charAt(a-1));

s2=String.*valueOf*(s2.charAt(a-1));

**if**(s1.equalsIgnoreCase(s2))

{

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

}

**else**

{

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

}

}

}

8.       **Month : Number of Days**

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

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

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

**Input and Output Format:**

Input consists of 2 integers that correspond to the year and month code.

Output consists of an integer that correspond to the number of days in the month in the given year.

**Sample Input:**

2000

1

**Sample Output:**

29

**package** gokul.javarevsi.dates;

**import**java.io.\*;

**import** java.text.ParseException;

**import**java.text.SimpleDateFormat;

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

**publicclass** gkdate {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

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

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

**int** days = 0;

**if** (month == 0 || month == 2 || month == 4 || month == 6 || month == 7

|| month == 9 || month == 11) {

days = 31;

} **elseif** (month == 3 || month == 5 || month == 8 || month == 10) {

days = 30;

} **elseif** (month == 1) {

GregorianCalendar gc = **new** GregorianCalendar();

**boolean** b = gc.isLeapYear(year);

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

days = 29;

} **else** {

days = 28;

}

}

System.*out*.println(days);

}

}

9.       **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

**import**java.io.\*;

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

**publicclass** Main {

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

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

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

System.*out*.println(usermaincode.*addOddNumbers*(s));

}

}

**import**java.util.ArrayList;

**import**java.util.Scanner;

**publicclass** usermaincode {

**publicstaticint** addOddNumbers(**int** s) {

**int** sum = 0;

**for** (**int** i = 1; i <= s; i++) {

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

sum += i;

}

}

**return** sum;

}

}

10.       **Even Sum & Duplicate Elements**

Write a program to read a integer array, Remove the duplicate elements and display sum of even numbers in the output. If input array contain only odd number then return -1.  
Include a class UserMainCode with a static method **sumElements** which accepts the integer array. The return type is integer.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
2  
3  
54  
1  
6  
7  
7  
**Sample Output 1:**  
62  
  
**Sample Input 2:**  
6  
3  
7  
9  
13  
17  
21  
**Sample Output 2:**  
-1

import java.io.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int s=sc.nextInt();

int sum=0,c=0,op=0;

int a[]=new int[s];

int b[]=new int[s];

for(int i=0;i<s;i++)

{

a[i]=sc.nextInt();

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

{

c++;

}

}

if(c==s)

{

op=-1;

System.out.println(op);

return;

}

for(int i=0;i<a.length;i++)

{

for(int j=i+1;j<a.length;j++)

{

if(a[i]!=a[j])

{

b[i]=a[i];

}

}

}

for(int i=0;i<b.length;i++)

{

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

{

sum+=b[i];

}

}

System.out.println(sum);

}

}

11.   **ArrayList to String Array**

Write a program that performs the following actions:  
    Read n strings as input.  
    Create an arraylist to store the above n strings in this arraylist.  
    Write a function convertToStringArray which accepts the arraylist as input.  
    The function should sort the elements (strings) present in the arraylist and convert them into a string array.  
    Return the array.  
Include a class UserMainCode with the static method **convertToStringArray** which accepts an arraylist and returns an array.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of n+1 integers. The first integer denotes the size of the arraylist, the next n strings are values to the arraylist.  
Output consists of an arrayas per step 4.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
a  
d  
c  
b  
**Sample Output 1:**  
a  
b  
c  
d

import java.util.ArrayList;

import java.util.Scanner;

public class G11{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

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

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

a.add(sc.next());

}

String s[]=UserMainCode.convertToStringArray(a);

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

System.out.println(s[i]);

}

}

}

import java.util.ArrayList;

import java.util.Collections;

public class UserMainCode {

public static String[] convertToStringArray(ArrayList<String> a){

Collections.sort(a);

String s[]=new String[a.size()];

a.toArray(s);

return s;

}

}

12.   **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

import java.io.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String s1=sc.next();

String s2=s1.replaceAll("[a-zA-Z]", "");

System.out.println(s2);

}

}

13.   **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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int x1 = sc.nextInt();

int y1 = sc.nextInt();

int x2 = sc.nextInt();

int y2 = sc.nextInt();

int res = (int) Math.round(Math

.sqrt((((x1 - x2) \* (x1 - x2)) + ((y1 - y2) \* (y1 - y2)))));

System.out.println(res);

}

}

14.   **Find common characters and unique characters in string**

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

Note:

- Space should not be counted as a letter.

- Consider letters to be case sensitive. ie, "a" is not equal to "A".

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

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

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

**Input and Output Format:**

Input consists of two strings.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

a black cow

battle ship

**Sample Output 1:**

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

**Sample Input 2:**

australia

sri lanka

**Sample Output 2:**

4

import java.util.Scanner;

public class G14 {

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);

StringBuffer sb2=new StringBuffer(s2);

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++;

}

}

}

System.out.println(count);

}

}

15.   **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

import java.util.Scanner;

public class G15 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String s[]=new String[n];

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

s[i]=sc.next();

}

StringBuffer sb=new StringBuffer();

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

int l=s[i].length()-1;

sb.append(s[i].charAt(l));

}

System.out.println(sb);

}

}

16.   **ArrayList to String Array**

Write a program that performs the following actions:  
  
1.Read m strings as input (fruit names).  
2.Create an arraylist to store the above m strings in this arraylist.  
3.Read n strings as input (fruit names).  
4.Create an arraylist to store the above n strings in this arraylist.  
5.Write a function fruitSelector which accepts the arraylists as input.  
6.Remove all fruits whose name ends with 'a' or 'e' from first arrayList and remove all fruits whose name begins  with 'm' or 'a' from second arrayList then combine the two lists and return the final output as a String array.  
7.If the array is empty the program will print as “No fruit found”  
Include a class UserMainCode with the static method **fruitSelector** which accepts the two arraylists and returns an array.  
  
Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.  
  
Output consists of an array as per step 6. Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
Apple  
Cherry  
Grapes  
4  
Orange  
Mango  
Melon  
Apple  
**Sample Output 1:**  
Cherry  
Grapes  
Orange

import java.util.ArrayList;

import java.util.Scanner;

public class G16 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int m=sc.nextInt();

ArrayList<String> al1=new ArrayList<String>();

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

al1.add(sc.next());

}

int n=sc.nextInt();

ArrayList<String> al2=new ArrayList<String>();

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

al2.add(sc.next());

}

String s[]=UserMainCode.fruitSelector(al1,al2);

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

System.out.println(s[i]);

}

}

}

import java.util.ArrayList;

public class UserMainCode {

public static String[] fruitSelector(ArrayList<String> al1,ArrayList<String> al2){

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

String k="";

for(int i=0;i<al1.size();i++){

k=al1.get(i);

int len=k.length()-1;

if(k.charAt(len)=='a' || k.charAt(len)=='e'){

continue;

}

else{

op.add(k);

}

}

for(int i=0;i<al2.size();i++){

k=al2.get(i);

if(k.charAt(0)=='a' || k.charAt(0)=='m'){

continue;

}

else{

op.add(k);

}

}

String s[]=new String[op.size()];

op.toArray(s);

return s;

}

}

**17. Elements in ArrayList**

Use Collection Methods.  
Write a program that takes two ArrayLists as input and  finds out all elements present either in A or B, but not in both.

Include a class UserMainCode with the static method arrayListSubtractor which accepts the two arraylists and returns an array.  
  
Create a Class Main which would be used to read the inputs and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.  
  
Output consists of an array. The elements in the output array need to be printed in sorted order.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
4  
1  
8  
3  
5  
2  
3  
5  
**Sample Output 1:**  
1  
8  
  
**Sample Input 2:**  
4  
9  
1  
3  
5  
4  
1  
3  
5  
6  
**Sample Output 2:**  
6  
9

import java.util.ArrayList;

import java.util.Scanner;

public class G17 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

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

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

int n=sc.nextInt();

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

A.add(sc.nextInt());

}

int m=sc.nextInt();

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

B.add(sc.nextInt());

}

int a[]=UserMainCode.arrayListSubtractor(A, B);

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

System.out.println(a[i]);

}

}

}

import java.util.ArrayList;

public class UserMainCode {

public static int[] arrayListSubtractor(ArrayList<Integer> A,ArrayList<Integer> B){

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

op.addAll(A);

op.removeAll(B);

B.removeAll(A);

op.addAll(B);

int a[]=new int[op.size()];

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

a[i]=op.get(i);

}

return a;

}

}

**18.  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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int sum = 0;

String s1 = sc.next();

StringBuffer sb = new StringBuffer();

String s2 = s1.replaceAll("[a-zA-Z]", "");

if (s2.isEmpty()) {

System.out.println(-1);

return;

}

else {

int a = Integer.parseInt(s2);

while (a != 0) {

int rem;

rem = a % 10;

sum += rem;

a /= 10;

}

System.out.println(sum);

}

}

}

**19.  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

import java.util.Scanner;

public class G19 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String s[]=new String[n];

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

s[i]=sc.next();

}

int a=sc.nextInt();

System.out.println(UserMainCode.countWord(s, a));

}

}

public class UserMainCode {

public static int countWord(String[] s,int a){

int count=0;

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

if(s[i].length()==a){

count++;

}

}

return count;

}

}

**20.  IP Validator**

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

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

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

**Input and Output Format:**

Input consists of a string that corresponds to an IP.

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

Refer sample output for formatting specifications.

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

**Sample Input 1:**

132.145.184.210

**Sample Output 1:**

Valid

**Sample Input 2:**

132.145.184.290

**Sample Output 2:**

Invalid

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int flag = 0;

String s1 = sc.next();

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

while (st.hasMoreTokens()) {

String s2 = st.nextToken();

int n = Integer.parseInt(s2);

if (n > 0 && n <= 255) {

flag++;

}

}

if (flag == 4) {

System.out.println("valid Ip");

} else {

System.out.println("Not a valid IP");

}

}

}

**21.  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

import java.util.Scanner;

public class G21 {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String s1=sc.nextLine();

String s2=sc.nextLine();

int temp=UserMainCode.getAnagram(s1, s2);

if(temp==1){

System.out.println("Anagrams");

}

else{

System.out.println("Not Anagrams");

}

}

}

import java.util.ArrayList;

import java.util.Collections;

public class UserMainCode {

public static int getAnagram(String s1,String s2){

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

ArrayList<Character> b=new ArrayList<Character>();

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

a.add(s1.charAt(i));

}

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

b.add(s2.charAt(i));

}

Collections.sort(a);

Collections.sort(b);

if(a.containsAll(b) || b.containsAll(a)){

return 1;

}

else{

return 0;

}

}

}

**22.  String processing – Long + Short + Long**

Obtain two strings S1,S2 from user as input. Your program should form a string of  “long+short+long”, with the shorter string inside of the longer String.  
Include a class UserMainCode with a static method **getCombo** which accepts two string variables. The return type is the string.  
Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.  
**Input and Output Format:**  
Input consists of two strings with maximum size of 100 characters.  
Output consists of an string.  
Refer sample output for formatting specifications.  
**Sample Input 1:**  
Hello  
Hi  
**Sample Output 1:**  
HelloHiHello

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

String s2 = sc.next();

int len1 = s1.length();

int len2 = s2.length();

StringBuffer sb = new StringBuffer(s1);

StringBuffer sb1 = new StringBuffer(s2);

if (len1 > len2) {

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

System.out.println(sb);

}

else {

sb1.append(s1).append(s2);

System.out.println(sb1);

}

}

}

**23.  Odd Digit Sum**

Write a program to input a String array. The input may contain digits and alphabets (“de5g4G7R”). Extract odd digits from each string and find the sum and print the output.

For example, if the string is "AKj375A" then take 3+7+5=15 and not as 375 as digit.

Include a class **UserMainCode** with a static method **oddDigitSum** which accepts a string array and the size of the array. The return type (Integer) should return the sum.

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

Assume maximum length of array is 20.

**Input and Output Format:**

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

Output consists of an Integer.

Refer sample output for formatting specifications.

**Sample Input :**

3

cog2nizant1

al33k

d2t4H3r5

**Sample Output :**

15

**(1+3+3+3+5)**

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int s = sc.nextInt();

int sum = 0;

String s1[] = new String[s];

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

s1[i] = sc.next();

}

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

System.out.println(s1[i]);

}

String s2[] = new String[s];

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

s2[i] = s1[i].replaceAll("[a-zA-Z]", "");

}

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

System.out.println(s2[i]);

}

int s3[] = new int[s];

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

s3[i] = Integer.parseInt(s2[i]);

}

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

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

al.add(s3[i]);

}

System.out.println(al);

Iterator<Integer> itr = al.iterator();

{

while (itr.hasNext()) {

int x = itr.next();

int rem;

while (x != 0) {

rem = x % 10;

if (rem % 2 != 0) {

sum += rem;

}

x /= 10;

}

}

System.out.println(sum);

}

}

}

**24.  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 Input2:**

this

1

**Sample Output 2:**

ts

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

int a = sc.nextInt();

StringBuffer sb = new StringBuffer();

int b = s1.length();

int c = b - a;

if (b > 2 \* a) {

sb.append(s1.substring(0, a)).append(s1.substring(c));

System.out.println(sb);

} else {

System.out.println("Error");

return;

}

}

}

**25.  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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int decimal\_val = 0, base = 1, rem;

while (n > 0) {

rem = n % 2;

decimal\_val = decimal\_val + rem \* base;

n /= 2;

base \*= 10;

}

System.out.println(decimal\_val);

}

}

**26.  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

import java.io.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

StringBuffer sb = new StringBuffer(s1);

String s2 = sb.reverse().toString();

if (s1.equals(s2)) {

System.out.println("Valid");

} else {

System.out.println("Not a valid");

}

}

}

**26.  States and Capitals**

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

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

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

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

**Sample Input 1:**

3

Karnataka

Bangaluru

Punjab

Chandigarh

Gujarat

Gandhinagar

Punjab

**Sample Output 1:**

chandigarh$punjab

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.Scanner;

**import** java.util.Set;

**public** **class** Main {

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

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

**int** a = sc.nextInt();

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

**for** (**int** i = 0; i < a; i++) {

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

}

System.*out*.println(hm);

String s1 = sc.next();

**for** (Map.Entry e : hm.entrySet()) {

**if** (e.getKey().equals(s1)) {

System.*out*.println(e.getValue() + "$" + e.getKey());

}

}

}

}

**27.  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

**28.  Leap Year**

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.  
  
Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The return type is the boolean indicating TRUE / FALSE.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string.  
  
Output consists of TRUE / FALSE.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
  
23/02/2012  
  
**Sample Output 1:**  
  
TRUE  
  
**Sample Input 2:**  
  
12/12/2011  
  
**Sample Output 2:**  
  
FALSE

**import** java.text.ParseException;

**import** java.util.\*;

**public** **class** Main {

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

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

String s = sc.next();

**boolean** b = **false**;

StringTokenizer st = **new** StringTokenizer(s, "/");

**while** (st.hasMoreTokens()) {

**int** day = Integer.*parseInt*(st.nextToken());

**int** month = Integer.*parseInt*(st.nextToken());

**int** year = Integer.*parseInt*(st.nextToken());

GregorianCalendar gc = **new** GregorianCalendar();

b = gc.isLeapYear(year);

System.*out*.println(b);

}

}

}

**29.  Vowel Check**

Write a program to read a String and check if that String contains all the vowels. Print “yes” if the string contains all vowels else print “no”.

Include a class **UserMainCode** with a static method **getVowels** which accepts a string. The return type (integer) should return 1 if the String contains all vowels else return -1.

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

**Input and Output Format:**

Input consists of a string.

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

Refer sample output for formatting specifications.

**Sample Input 1:**

abceiduosp

**Sample Output 1:**

yes

**Sample Input 2:**

bceiduosp

**Sample Output 2:**

no

import java.text.ParseException;

import java.util.\*;

public class Main {

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

Scanner sc = new Scanner(System.in);

String s = sc.next();

String s2 = s.replaceAll("[^aeiouAEIOU]", "");

System.out.println(s2);

HashSet<Character> hs = new HashSet<Character>();

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

hs.add(s2.charAt(i));

}

if (hs.size() == 5) {

System.out.println("yes");

} else {

System.out.println("No");

}

}

}

**0r**

**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 s=sc.nextLine();

String v="aeiou";

**int** count=0;

**for**(**int** i=0;i<s.length();i++)

{

**for**(**int** j=0;j<v.length();j++)

{

**if**(v.charAt(j)==s.charAt(i))

{

count=1;

}

}

}

**if**(count==1)

{

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

}

**else**

{

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

}

}

}

**30.**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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

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));

}

}

}

}

System.out.println(sb1.toString());

}

}

**31.  Largest Element**

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

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

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

Assume maximum length of array is 20.

**Input and Output Format:**

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

Output consists of a single Integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

2

3

8

4

5

**Sample Output 1:**

8

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int first = 0, mid = 0, last = 0;

int s = sc.nextInt();

int a[] = new int[s];

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

a[i] = sc.nextInt();

}

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

first = a[0];

mid = a[(s - 1) / 2];

last = a[s - 1];

}

if (first > mid) {

System.out.println(first);

} else if (mid > last) {

System.out.println(mid);

} else if (last > first) {

System.out.println(last);

}

}

}

**32.  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  
  
import java.text.ParseException;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

import java.util.StringTokenizer;

import java.util.TreeMap;

public class Sum {

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

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String s[]=new String[n];

String ss="01-09-2014";

StringTokenizer st=new StringTokenizer(ss,"-");

int ds=0,ms=0,ys=0;

while(st.hasMoreTokens()){

ds=Integer.parseInt(st.nextToken());

ms=Integer.parseInt(st.nextToken());

ys=Integer.parseInt(st.nextToken());

}

int ids[]=new int[n];

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

HashMap<Integer,Integer> h2=new HashMap<Integer, Integer>();

TreeMap<Integer,Integer> t1=new TreeMap<Integer, Integer>();

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

int id=sc.nextInt();

ids[i]=id;

s[i]=sc.next();

int sal=sc.nextInt();

h1.put(id,s[i]);

h2.put(id,sal);

}

int d=0,y=0,m=0,sals=0;

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

StringTokenizer st1=new StringTokenizer(s[i],"-");

while(st1.hasMoreTokens()){

d=Integer.parseInt(st1.nextToken());

m=Integer.parseInt(st1.nextToken());

y=Integer.parseInt(st1.nextToken());

}

int age=0;

if((d<ds || m<ms) && y==ys){

age=(ys-y)-1;

}

else{

age=ys-y;

}

System.out.println(age);

if(age>25 && age<=30){

sals=h2.get(ids[i]);

sals=sals+sals/5;

}

else if(age>30 && age<=60){

sals=h2.get(ids[i]);

sals=sals+((sals\*3)/10);

}

t1.put(ids[i],sals);

}

for(Map.Entry<Integer,Integer> e:t1.entrySet()){

System.out.println(e.getKey()+"\n"+e.getValue());

}

}

}

**33.  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

**import** java.text.ParseException;

**import** java.util.\*;

**public** **class** Main {

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

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

String s = sc.next();

**if** (s.matches("((?=.\*[0-9])(?=.\*[a-zA-Z])(?=.\*[@#$!]).{8,})")) {

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

} **else** {

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

}

}

}

**34.  Length of same word**

Write a program to read a string containing multiple words find the first and last words, if they are same, return the length and if not return the sum of length of the two words.  
  
Include a class UserMainCode with a static method **compareLastWords** which accepts the string. The return type is the length as per problem.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
This is Cognizant Academy  
  
**Sample Output 1:**  
11  
  
**Sample Input 2:**  
Hello World Hello  
  
**Sample Output 2:**  
5

**import** java.text.ParseException;

**import** java.util.\*;

**public** **class** Main {

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

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

String s1 = sc.nextLine();

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

String s2 = st.nextToken();

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

sb.reverse();

String s3 = sb.toString();

StringTokenizer st11 = **new** StringTokenizer(s3, " ");

String x = st11.nextToken();

StringBuffer sb1 = **new** StringBuffer(x);

sb1.reverse();

String s4 = sb1.toString();

**if** (s2.equalsIgnoreCase(s4))

System.*out*.println(s2.length());

**else**

System.*out*.println(s2.length() + x.length());

}

}

**35.  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

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** Main {

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

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

**int** s = sc.nextInt();

**int** a[] = **new** **int**[s];

**int** mid;

**for** (**int** i = 0; i < s; i++) {

a[i] = sc.nextInt();

}

Arrays.*sort*(a);

**if** (s % 2 != 0) {

mid = a[(s - 1) / 2];

} **else** {

mid = Math.*round*((a[s / 2] + a[(s / 2) - 1]) / 2);

}

System.*out*.println(mid);

}

}

**36.  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

**import** java.util.Arrays;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

String s1 = sc.nextLine();

String s3 = sc.next();

**int** count = 0;

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

**while** (st.hasMoreElements()) {

String s2 = st.nextToken();

**if** (s2.equals(s3)) {

count++;

}

}

System.*out*.println(count);

}

}

**37.  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

**import** java.util.Arrays;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

String s1 = sc.nextLine();

**int** a = sc.nextInt();

**int** b = sc.nextInt();

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

sb.reverse();

StringBuffer sb1 = **new** StringBuffer();

String ss = sb1.append(sb.substring(a, a + b)).toString();

System.*out*.println(ss);

}

}

**38.  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

**package** gokul.javarevsi.dates;

**import**java.io.\*;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

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

**publicclass** gkdate {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

String s1 = sc.next();

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

sdf.setLenient(**false**);

Date d1 = sdf.parse(s1);

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

String s2 = sdf1.format(d1);

System.*out*.println(s2.toUpperCase());

}

}

**39.  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**

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

**int** a = sc.nextInt();

ArrayList<Integer> al1 = **new** ArrayList<Integer>();

ArrayList<Integer> al2 = **new** ArrayList<Integer>();

ArrayList<Integer> al3 = **new** ArrayList<Integer>();

**for** (**int** i = 0; i < a; i++) {

al1.add(sc.nextInt());

}

**for** (**int** i = 0; i < a; i++) {

al2.add(sc.nextInt());

}

al1.addAll(al2);

System.*out*.println(al1);

Collections.*sort*(al1);

System.*out*.println(al1);

**for** (**int** i = 0; i < al1.size(); i++) {

**if** (i == 2 || i == 6 || i == 8) {

al3.add(al1.get(i));

}

}

System.*out*.println(al3);

}

}

**40.  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](https://mail.cognizant.com/owa/redir.aspx?SURL=Wln5vWbypjQm7khVJZKF5lnNio6OpE6bmlopUbW4b64qMbBgHMTSCG0AYQBpAGwAdABvADoAYQBkAG0AaQBuAEAAeAB5AHoALgBjAG8AbQA.&URL=mailto%3aadmin%40xyz.com)  
  
**Sample Output 1:**  
admin

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

String s1 = sc.nextLine();

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

String name = st.nextToken();

System.*out*.println(name);

}

}

**41.  ID Validation**

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

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

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

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

**Input and Output Format:**

Input consists of two strings.

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

Output is a string Valid id or Invalid id.

Refer sample output for formatting specifications.

**Sample Input 1:**

CTS-hyd-1234

hyderabad

**Sample Output 1:**

Valid id

**Sample Input 2:**

CTS-hyd-123

hyderabad

**Sample Output 2:**

Invalid id

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

String s1 = sc.nextLine();

String s2 = sc.next();

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

**while** (st.hasMoreElements()) {

String id = st.nextToken();

String loc = st.nextToken();

String xxx = st.nextToken();

**if** (loc.matches(s2.substring(0, 3)) && xxx.length() == 4) {

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

} **else** {

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

}

}

}

}

**42.  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

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

**int** s = sc.nextInt();

**int** sum = 0, avg = 0, count = 0, total;

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

**for** (**int** i = 0; i < s; i++) {

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

}

System.*out*.println(hm);

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

{

**while** (itr.hasNext()) {

**int** j = itr.next();

**if** (j % 2 != 0) {

sum += hm.get(j);

count++;

}

}

total = sum / count;

System.*out*.println(total);

}

}

}

**43.  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

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.HashMap;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

String s = sc.nextLine();

String s2 = s.replaceAll("[^aeiouAEIOU]", "");

System.*out*.println(s2);

HashSet<Character> hs = **new** HashSet<Character>();

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

hs.add(s2.charAt(i));

}

System.*out*.println(hs);

**if** (hs.size() == 5) {

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

} **else** {

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

}

}

}

**44.  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

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public** **class** Main {

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

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

String s1 = sc.next();

**if** (s1.matches

("((?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.\*[@#$!%]).{8,})"))

{

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

} **else** {

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

}

}

}

**45.  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

import java.util.Arrays;

import java.util.Collections;

import java.util.HashMap;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int flag = 0, sum = 0;

double tot = 0, count = 0.0;

int a[] = new int[8];

for (int i = 0; i < 8; i++) {

a[i] = sc.nextInt();

}

for (int i = 0; i < 8; i++) {

flag = 0;

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

if (i % j == 0) {

flag++;

}

}

if (flag == 2) {

sum = sum + a[i];

count++;

}

}

tot = (double) (sum / count);

System.out.println(tot);

}

}

**46.  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

import java.util.Scanner;

public class G46{

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();

}

if(a.length%2!=0){

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

}

else{

System.out.println("enter the odd number of elemnts");

}

}

}

public class UserMainCode {

public static int getMiddleElement(int[] a){

int count=a.length;

return a[count/2];

}

}

**47.  Negative String**

Given a string input, write a program to replace every appearance of the word "is" by "is not".

If the word "is" is immediately preceeded or followed by a letter no change should be made to the string .

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

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input 1:**

This is just a misconception

**Sample Output 1:**

This is not just a misconception

**Sample Input 2:**

Today is misty

**Sample Output 2:**

Today is not misty

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

String s2 = s1.replace(" is", " is not");

System.out.println(s2);

}

}

**48.  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

import java.util.Scanner;

public class G48 {

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 sum=UserMainCode.getSumOfIntersection(a, b);

if(sum==0){

System.out.println("No Common Elements");

}

else{

System.out.println(sum);

}

}

}

public class UserMainCode {

public static int getSumOfIntersection(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+b[j];

}

}

}

return sum;

}

}

**49.  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

**import**java.io.\*;

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

**publicclass** Main {

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

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

String s1 = sc.next();

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

String s2 = sb.append(s1.substring(0, 1)).toString();

System.*out*.println(s2);

**if** (s2.matches("[a-z]{1}")) {

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

} **else** {

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

}

}

}

**50.  Largest Chunk**

Write a program to read a string and return the length of the largest "chunk" in the string.  
A chunk is a repetition of same character 2 or more number of times. If the given string doest not contain any repeated chunk of characters return -1.  
Include a class UserMainCode with a static method **getLargestSpan** which accepts the string. The return type is the integer.  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of a string.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
This place is soooo good  
**Sample Output 1:**  
4

import java.util.Scanner;

public class G50{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

if(UserMainCode.getLargestSpan(s)==-1){

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

}

else{

System.out.println(UserMainCode.getLargestSpan(s));

}

}

}

import java.util.StringTokenizer;

public class UserMainCode {

public static int getLargestSpan(String a){

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

int max=0;

while(st.hasMoreTokens()){

String s=st.nextToken();

StringBuffer sb=new StringBuffer(s);

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)){

count++;

}

}

if(count>max){

max=count+1;

}

}

}

if(max==0){

return -1;

}

else{

return max;

}

}

}

**51.  Find Digits**

For a given double number with atleast one decimal value, Write a program to compute the number of digits before and after the decimal point in the following format – noOfDigitsBeforeDecimal:noOfDigitsAfterDecimal.  
  
Note: Ignore zeroes at the end of the decimal (Except if zero is the only digit after decimal. Refer Example 2 and 3)  
  
Include a class UserMainCode with a static method **findNoDigits** which accepts the decimal value. The return type is string.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a double.  
Output consists of string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
843.21  
  
**Sample Output 1:**  
3:2  
  
**Sample Input 2:**  
20.130  
  
**Sample Output 2:**  
2:2  
  
**Sample Input 3:**  
20.130  
  
**Sample Output 3:**  
2:2

import java.io.\*;

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

double a = sc.nextDouble();

String b = String.valueOf(a);

StringBuffer sb = new StringBuffer();

StringTokenizer st = new StringTokenizer(b, ".");

String c = st.nextToken();

String d = st.nextToken();

System.out.println(c);

System.out.println(d);

int x = c.length();

int y = d.length();

sb.append(x).append(':').append(y);

System.out.println(sb);

}

}

**52.  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

import java.io.\*;

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String a = sc.next();

int s = sc.nextInt();

String b = "";

StringBuffer sb = new StringBuffer(a);

if (s >= 3) {

b = sb.substring(0, 3);

} else if (s <= 2) {

b = sb.substring(0, s);

}

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

System.out.print(b + " ");

}

}

}

**53.  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.io.\*;

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int a = sc.nextInt();

int x = 0, y = 0, z = 0;

int b = a \* a;

String c = String.valueOf(b);

int d = c.length();

if (d == 2) {

x = b % 10;

y = b / 10;

z = x + y;

}

else if (d == 4) {

x = b % 100;

y = b / 100;

z = x + y;

}

System.out.println(z);

if (z == a) {

System.out.println("Kaperakar number");

} else {

System.out.println("Not a Kaperakar Number");

}

}

}

**54.  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!

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

StringBuffer sb = new StringBuffer();

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

while (st.hasMoreElements()) {

String s2 = st.nextToken();

s2 = s2.substring(0, 1).toUpperCase() + s2.substring(1) + " ";

sb.append(s2);

}

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

}

}

**55.  String Concatenation**

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

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

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

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

**Input and Output Format:**

Input consists of two strings.

Output is a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

Hello

hi

**Sample Output 1:**

lohi

**Sample Input 2:**

Hello

Delhi

**Sample Output 2:**

HelloDelhi

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

String s2 = sc.nextLine();

int a1 = s1.length();

int a2 = s2.length();

StringBuffer sb = new StringBuffer(s1);

if (a1 == a2) {

System.out.println(s1.concat(s2));

} else {

sb.reverse();

String s3 = sb.substring(0, a2);

StringBuffer sb1 = new StringBuffer(s3);

sb1.reverse().append(s2);

System.out.println(sb1.toString());

}

}

}

**56.  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

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

int count = 0;

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

while (st.hasMoreElements()) {

String s2 = st.nextToken();

count++;

}

System.out.println(count);

}

}

**57.  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

import java.util.Scanner;

public class G57{

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.checkDifference(a));

}

}

public class UserMainCode {

public static int checkDifference(int[] a){

int max=0,p=0;

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

int j=i+1;

int t=a[i]-a[j];

if(t>max){

max=t;

p=i;

}

}

return p;

}

}

**58.  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.io.\*;

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

**publicclass** Main {

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

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

String s1 = sc.next();

**if** (s1.matches("([1-12]{2})/([1-31]{2})/([0-9]{4})")) {

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

} **else** {

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

}

}

}

**59.  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

**import**java.io.\*;

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

**publicclass** Main {

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

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

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

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

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

**int** sum = 0;

**if** (a == b) {

sum = c;

} **elseif** (b == c) {

sum = a;

} **elseif** (c == a) {

sum = b;

} **else** {

sum = a + b + c;

}

System.*out*.println(sum);

}

}

**60.  Initial Format**

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

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

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

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

**Sample Input :**

Jessica Miller

**Sample Output:**

Miller, J

**import**java.io.\*;

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

**publicclass** Main {

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

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

String s = sc.nextLine();

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

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

**while** (st.hasMoreElements()) {

String s1 = st.nextToken();

String s2 = st.nextToken();

sb.append(s2).append(", ").append(s1.substring(0, 1));

}

System.*out*.println(sb);

}

}

**61.  Remove Elements**

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

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

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

Assume maximum length of array is 20.

**Input and Output Format:**

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

Output consists of a single Integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

a

bb

b

ccc

ddd

2

**Sample Output 1:**

4

import java.util.\*;

public class ClassSet28 {

public static int StringsNotOfGivenLength(List<String> l1,String s1){

int n1=s1.length();

int c=0;

for(int i=0;i<l1.size();i++){

int n2=l1.get(i).length();

if(n1!=n2)

c++;

}

return c;

}

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

System.out.println("enter the no.of elements:");

int n=s.nextInt();

List<String> l1=new ArrayList<String>();

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

l1.add(s.next());

System.out.println("enter the input string:");

String s1=s.next();

System.out.println(StringsNotOfGivenLength(l1,s1));

}

}

**63.  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

**publicclass** Main {

**publicstatic** String reversedAndParsedString(String s1,**char**c){

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

StringBuffer sb1=**new** StringBuffer(s1);

sb.reverse();

**for**(**int** i=0;i<(2\*s1.length())-1;i++)

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

sb1=sb.insert(i,c);

**return** sb1.toString();

}

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

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

System.*out*.println("enter the String:");

String s1=s.next();

**char** c=s.next().charAt(0);

System.*out*.println("the formatted string is:"+*reversedAndParsedString*(s1,c));

}

}

**64.  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

**import**java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import**java.util.List;

**import** java.util.Scanner;

**publicclass** Main {

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

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

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

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

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

{

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

}

System.*out*.println(*getvalues*(hm));

}

**publicstatic** String getvalues(HashMap<Integer, String> hm) {

**int** b=0,max=0;

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

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

**while**(i.hasNext())

{

b=i.next();

**if**(b>max)

{

max=b;

s1=hm.get(b);

}

}

**return** s1;

}

}

**65.  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

import java.io.\*;

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();

}

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

if (a[i] == a[i + 1]) {

System.out.println("true");

return;

}

else if (a[i + 1] == a[i + 2]) {

System.out.println("True");

return;

}

else {

System.out.println("False");

return;

}

}

}

}

**66.  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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

if (s1.length() % 2 == 0) {

if (s1.length() > 2) {

System.out.println(s1.substring(s1.length() / 2 - 1,

s1.length() / 2 + 1));

}

}

}

}

**67.**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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

if (s1.matches("((?=.\*[0-9])(?=.\*[#@$])(?=.\*[a-z]).{6,20})")) {

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

} else {

System.out.println("Not a Valid PAssword");

}

}

}

**68.  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  
**Sample Output 2:**  
A Rope Ends It

**69.  Pattern Matcher**

Write a program to read a string and check if it complies to the pattern 'CPT-XXXXXX' where XXXXXX is a 6 digit number. If the pattern is followed, then print TRUE else print FALSE.  
  
Include a class UserMainCode with a static method **CheckID** which accepts the string. The return type is a boolean value.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string.  
  
Output should print TRUE or FALSE .  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
CPT-302020  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
CPT123412  
  
**Sample Output 2:**  
FALSE

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

if (s1.matches("[CPT-]{4}[0-9]{6}")) {

System.out.println("True");

} else {

System.out.println("False");

}

}

}

**70.  Max Admissions**

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

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

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

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

**Sample Input 1:**

4

2010

200000

2011

300000

2012

45000

2013

25000

**Sample Output 1:**

2011

**import**java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import**java.util.List;

**import** java.util.Scanner;

**publicclass** Main {

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

// **TODO** Auto-generated method stub

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

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

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

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

{

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

}

System.*out*.println(*getvalues*(hm));

}

**publicstaticint** getvalues(HashMap<Integer,Integer> hm) {

**int** b=0,max=0,c=0,d=0;

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

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

**while**(i.hasNext())

{

b=i.next();

c=hm.get(b);

**if**(c>max)

{

max=c;

d=b;

}

}

**return** d;

}

}

**71.  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.\*;

public class ClassSeT23 {

public static void main(String[] args) {

Map<String, Integer> m1=new HashMap<String, Integer>();

m1.put("abc", 90);

m1.put("efg", 50);

m1.put("mno", 60);

m1.put("rst", 75);

m1.put("xyz", 35);

System.out.println(examResult(m1));

}

public static Map<String,String> examResult(Map<String, Integer> m1) {

Map<String,String> m2=new HashMap<String, String>();

String s1=new String();

String s2=new String();

int n=0;

Iterator<String> i=m1.keySet().iterator();

while(i.hasNext()){

s1=(String) i.next();

n=m1.get(s1);

if(n>=60)

s2="PASS";

else

s2="FAIL";

m2.put(s1, s2); }

return m2;

}

}

**72.  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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

String s2 = s1.replaceAll("[aeiou]", "");

System.out.println(s1.length() - s2.length());

}

}

**73.  Validate Number**

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

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

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

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input 1:**

-94923

**Sample Output 1:**

94923

**Sample Input 2:**

-6t

**Sample Output 2:**

-1

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

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

int s2 = Math.abs(Integer.parseInt(s1));

String s3 = String.valueOf(s2);

System.out.println(s3);

} else {

System.out.println("-1");

}

}

}

**74.  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

**package** gokul.javarevsi.dates;

**import**java.io.\*;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

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

**publicclass** gkdate {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

String s1 = sc.next();

String s2 = sc.next();

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

// first String Date

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

sdf.setLenient(**false**);

Date d1 = sdf.parse(s1);

Calendar cal = Calendar.*getInstance*();

cal.setTime(d1);

Date d2 = cal.getTime();

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

**int** y1 = Integer.*parseInt*(sdf1.format(d2));

// Second String date

SimpleDateFormat sdf2 = **new** SimpleDateFormat("dd/MM/yyyy");

sdf2.setLenient(**false**);

Date d3 = sdf.parse(s2);

Calendar cal1 = Calendar.*getInstance*();

cal1.setTime(d3);

Date d4 = cal1.getTime();

SimpleDateFormat sdf3 = **new** SimpleDateFormat("yyyy");

**int** y2 = Integer.*parseInt*(sdf3.format(d4));

**int** exp = y2 - y1;

// boolean b=false;

**if** (exp == val) {

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

} **else** {

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

}

}

}

**75.  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]

**76.  Repeating set of characters in a string**

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

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

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

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

**Input and Output Format:**

Input consists a string and a positive integer n.

Output is a string with repeated characters.

Refer sample output for formatting specifications.

**Sample Input 1:**

Cognizant

3

**Sample Output 1:**

Cognizantantantant

**Sample Input 2:**

myacademy

2

**Sample Output 2:**

myacademymymy

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

int s2 = sc.nextInt();

int s3 = s1.length();

String s4 = s1.substring(s3 - s2);

System.out.println(s4);

StringBuffer sb = new StringBuffer(s1);

for (int i = 0; i < s2; i++) {

sb.append(s4);

}

System.out.println(sb);

}

}

**77.  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 Forma:**

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

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int a = sc.nextInt();

int rem, sum = 0, square;

while (a != 0) {

rem = a % 10;

if (rem % 2 == 0) {

square = rem \* rem;

sum += square;

}

a /= 10;

}

System.out.println(sum);

}

}

**78.  Regular Expression - 1**

Given a string (s) apply the following rules.  
  
1. String should be only four characters long.  
2. First character can be an alphabet or digit.  
3. Second character must be uppercase 'R'.  
4. Third character must be a number between 0-9.  
  
If all the conditions are satisifed then print TRUE else print FALSE.  
  
Include a class UserMainCode with a static method **validate** which accepts the string. The return type is the boolean formed based on rules.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string.  
  
Output consists of TRUE or FALSE .  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
vR4u  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
vRau  
  
**Sample Output 2:**  
FALSE  
  
**Sample Input 3:**  
vrau  
  
**Sample Output 3:**  
FALSE

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

if (s1.matches("[a-zA-Z0-9]{1}[R]{1}[0-9]{1}[a-zA-Z]{1}")) {

System.out.println("True");

} else {

System.out.println("False");

}

}

}

**79.  Reversing a Number**

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

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

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

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

**Input and Output Format:**

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

**Sample Input 1:**

543

**Sample Output 1:**

345

**Sample Input 1:**

1111

**Sample Output 1:**

1111

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int a = sc.nextInt();

int rem, rev = 0;

while (a != 0) {

rem = a % 10;

rev = rev \* 10 + rem;

a /= 10;

}

System.out.println(rev);

}

}

**80.  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

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();

Arrays.sort(a);

int sum = a[0] + a[s - 1];

float avg = (float) sum / 2;

System.out.println(avg);

}

}

**81.  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

**82.  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

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];

int count = 0;

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

a[i] = sc.nextInt();

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

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

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

count = i + j + 1;

}

}

}

System.out.println(count);

}

}

**83.  Sum Squares of Digits**

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

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

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

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

321

**Sample Output:**

14

import java.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 rem, num = 1, sum = 0;

while (s != 0) {

rem = s % 10;

num = rem \* rem;

sum += num;

s /= 10;

}

System.out.println(sum);

}

}

**84. 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

import java.util.\*;

public class ClassSeT40 {

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

String s1=s.next();

boolean b=passwordValidation(s1);

if(b==true)

System.out.println("valid password");

else

System.out.println("not a valid password");

}

public static boolean passwordValidation(String s1) {

boolean b=false,b1=false,b2=false;

if(s1.length()>=8)

if(!Character.isDigit(s1.charAt(0)))

if(s1.charAt(0)!='@' && s1.charAt(0)!='\_' && s1.charAt(0)!='#')

if(s1.charAt(s1.length()-1)!='@' && s1.charAt(s1.length()-1)!='\_' && s1.charAt(s1.length()-1)!='#')

b1=true;

if(b1==true)

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

if(Character.isAlphabetic(s1.charAt(i)) || Character.isDigit(s1.charAt(i)) || s1.charAt(i)=='#' || s1.charAt(i)=='@' || s1.charAt(i)=='\_')

b2=true;

if(b2==true)

if(s1.contains("#") || s1.contains("@") || s1.contains("\_"))

b=true;

return b;

}

}

**85.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

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int s = sc.nextInt();

int squaresum = 0, cubesum = 0;

int a[] = new int[s];

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

a[i] = sc.nextInt();

}

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

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

cubesum += a[i] \* a[i] \* a[i];

} else {

squaresum += a[i] \* a[i];

}

}

int cs = cubesum + squaresum;

System.out.println(cs);

}

}

**86.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

**87.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

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.next();

int a = sc.nextInt();

StringBuffer sb = new StringBuffer();

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

sb.append(s.substring(s.length() - a));

}

System.out.println(sb);

}

}

**88.String Processing - III**

Write a program to read a string where all the lowercase 'x' chars have been moved to the end of the string.  
  
Include a class UserMainCode with a static method **moveX** which accepts the string. The return type is the modified string.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string.  
Output consists of a string.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
xxhixx  
  
**Sample Output 1:**  
hixxxx  
  
**Sample Input 2:**  
XXxxtest  
  
**Sample Output 2:**  
XXtestxx

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.next();

String s1 = s.replaceAll("[x]", "");

String s2 = s.replaceAll("[^x]", "");

System.out.println(s1);

System.out.println(s2);

System.out.println(s1 + s2);

}

}

**89.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

89.

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String word = sc.nextLine();

String result = new String("");

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

if (!result.contains("" + word.charAt(i))) {

result += "" + word.charAt(i);

}

}

System.out.println(result);

}

}

**90.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

**package** gokul.javarevsi.dates;

**import**java.io.\*;

**import** java.text.ParseException;

**import**java.text.SimpleDateFormat;

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

**publicclass** gkdate {

**publicstaticvoid** main(String[] args) **throws** ParseException {

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

**int** i = 0, j = 0, count = 0;

String s1 = sc.next();

String s2 = sc.next();

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

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

**int** l1[] = **newint**[a];

**int** l2[] = **newint**[b];

StringTokenizer st1 = **new** StringTokenizer(s1, "-");

StringTokenizer st2 = **new** StringTokenizer(s2, "-");

**while** (st1.hasMoreElements()) {

String x = st1.nextToken();

l1[i] = x.length();

i++;

}

**while** (st2.hasMoreElements()) {

String x2 = st2.nextToken();

l2[j] = x2.length();

j++;

}

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

**if** (l1[c] == l2[c]) {

count = 1;

} **else** {

count = 0;

**break**;

}

}

**if** (count == 1)

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

**elseif** (count == 0)

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

}

}

**91.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.]

**92.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

import java.text.ParseException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

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();

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--;

}

}

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

}

}

**93.Even and Odd Index Sum**

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

Example:

input = 23050

evenSum = 2 + 0 + 0 = 2

oddSum = 3 + 5 = 8

output = no

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

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

**Input and Output Format:**

Input consists of an integer.

Output consists of a string that is either “yes” or “no”.

**Sample Input 1:**

23050

**Sample Output 1:**

no

**Sample Input 2:**

231

**Sample Output 2:**

yes

import java.text.ParseException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import java.util.Scanner;

public class Main {

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

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int rem, count = 0, esum = 0, osum = 0;

while (n != 0) {

rem = n % 10;

if (count % 2 == 0) {

esum += rem;

} else {

osum += rem;

}

count++;

n /= 10;

}

if (esum == osum) {

System.out.println("Yes");

} else {

System.out.println("No");

}

}

}

**94.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.text.ParseException;

import java.util.ArrayList;

import java.util.Collection;

import java.util.Collections;

import java.util.Scanner;

public class Main {

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

Scanner sc = new Scanner(System.in);

int s = sc.nextInt();

String s1[] = new String[s];

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

s1[i] = sc.next();

}

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

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

al.add(s1[i]);

}

Collections.sort(al);

String s3[] = new String[al.size()];

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

s3[i] = al.get(i).toLowerCase();

System.out.println(s3[i]);

}

}

}

----------------0r----------------------

import java.text.ParseException;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collection;

import java.util.Collections;

import java.util.Scanner;

public class Main {

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

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

String s2[] = new String[n];

String s1[] = new String[n];

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

s1[i] = sc.next();

s2[i] = s1[i].toLowerCase();

}

Arrays.sort(s2);

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

System.out.println(s2[i]);

}

}

}

**95.Find the difference between Dates in months**

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

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

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

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

**Input and Output Format:**

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

2012-03-01

2012-04-16

**Sample Output 1:**

1

**Sample Input 2:**

2011-03-01

2012-04-16

**Sample Output 2:**

13

import java.text.SimpleDateFormat;

import java.util.Calendar;

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();

String s2 = sc.nextLine();

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

sdf.setLenient(false);

Date d = sdf.parse(s1);

Date d1 = sdf.parse(s2);

Calendar cal = Calendar.getInstance();

cal.setTime(d);

int m1 = cal.get(Calendar.MONTH);

cal.setTime(d1);

int m2 = cal.get(Calendar.MONTH);

int res = Math.abs(m2 - m1);

System.out.println(res);

}

}

**96.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

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.next();

String s2 = "abcdefghijklmnopqrstuvwxyza";

StringBuffer sb = new StringBuffer();

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

if (i % 2 != 0) {

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

} else {

int n = s2.indexOf(s1.charAt(i));

sb.append(s2.charAt(n + 1));

}

}

System.out.println(sb);

}

}

**97.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

import java.text.ParseException;

import java.text.SimpleDateFormat;

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 < a.length; i++) {

a[i] = sc.nextInt();

}

if (a[0] == 9 || a[1] == 9 || a[2] == 9 || a[3] == 9) {

System.out.println("true");

} else {

System.out.println("False");

}

}

}

**98.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

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

int max = 0;

String s4 = "";

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

while (st.hasMoreTokens()) {

String s2 = st.nextToken();

String s3 = s2.replaceAll("[aeiouAEIOU]", "");

int len = s3.length();

if (len > max) {

max = len;

s4 = s2;

}

}

System.out.println(s4);

}

}

**99.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

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

boolean b = false;

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

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

sdf.setLenient(false);

try {

Date d = sdf.parse(s1);

System.out.println("Valid");

} catch (ParseException e) {

System.out.println("Invalid");

}

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

SimpleDateFormat sdf1 = new SimpleDateFormat("dd.MM.yyyy");

sdf1.setLenient(false);

try {

Date d1 = sdf1.parse(s1);

System.out.println("Valid");

} catch (ParseException e) {

System.out.println("Invalid");

}

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

SimpleDateFormat sdf2 = new SimpleDateFormat("dd-MM-yyyy");

sdf2.setLenient(false);

try {

Date d2 = sdf2.parse(s1);

System.out.println("Valid");

} catch (ParseException e) {

System.out.println("Invalid");

}

} else {

System.out.println("Invalid");

}

}

}

**100.Phone Number Validator**

Given a phone number as a string input, write a program to verify whether the phone number is valid using the following business rules:

-It should contain only numbers or dashes (-)

- dashes may appear at any position

-Should have exactly 10 digits

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

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

**Input and Output Format:**

Input consists of a string.

Output consists of a string that is either 'Valid' or 'Invalid'

**Sample Input 1:**

265-265-7777

**Sample Output 1:**

Valid

**Sample Input 2:**

265-65-7777

**Sample Output 1:**

Invalid

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

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

Scanner sc = new Scanner(System.in);

String s1 = sc.nextLine();

if (s1.matches("[0-9-]{12}")) {

System.out.println("Valid");

} else {

System.out.println("Invalid");

}

}

}

**101.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.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

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

Scanner sc = new Scanner(System.in);

int s = sc.nextInt();

int a[] = new int[20];

int flag = 0, sum = 0, c = 0, j;

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

a[i] = sc.nextInt();

}

for (int i = 2; i < a.length; i++) {

for (j = 2; j < i; j++) {

if (i % j == 0) {

c++;

}

}

if (c == 0) {

sum += a[i];

}

}

int avg = sum / 2;

System.out.println(sum);

System.out.println(avg);

}

}

**102.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)**

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.\*;

public class Main {

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

Scanner sc = new Scanner(System.in);

int ll = sc.nextInt();

int ul = sc.nextInt();

int rem, sum = 0, rev = 0, temp = 0;

for (int i = ll; i < ul; i++) {

rev = 0;

temp = i;

while (temp != 0) {

rem = temp % 10;

rev = rev \* 10 + rem;

temp /= 10;

}

if (rev == i) {

sum += rev;

}

}

System.out.println(sum);

}

}

**103.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

**import**java.io.\*;

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

**publicclass** Main {

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

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

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

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

**int** n2 = 0;

**char** c = sc.next().charAt(0);

**if** (c == '+') {

n2 = n + n1;

System.*out*.println(n2);

} **elseif** (c == '-') {

n2 = n - n1;

System.*out*.println(n2);

} **elseif** (c == '\*') {

n2 = n \* n1;

System.*out*.println(n2);

} **elseif** (c == '/') {

n2 = n / n1;

System.*out*.println(n2);

} **elseif** (c == '%') {

n2 = n % n1;

System.*out*.println(n2);

} **else**

System.*out*.println(-1);

}

}

**104.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

**import**java.io.\*;

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

**publicclass** Main {

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

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

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

**int** i, k = 0;

**int** a[] = **newint**[n];

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

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

a[i] = sc.nextInt();

}

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

**if** (a[i] != 5) {

al.add(a[i]);

}

}

**if** (al.size() < n) {

k = n - al.size();

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

al.add(0);

}

}

**int** b[] = **newint**[al.size()];

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

b[i] = al.get(i);

System.*out*.println(b[i]);

}

}

}

**105.Repeat Front**

Given a string (s) and non negative integer (n) apply the following rules.

1.         Display the first three characters as front.

2.         If the length of the string is less than 3, then consider the entire string as front and repeat it n times.

Include a class UserMainCode with a static method **repeatFirstThreeCharacters** which accepts the string and integer. The return type is the string formed based on rules.  
  
Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string and integer.  
  
Output consists of a string .  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
Coward  
2  
  
**Sample Output 1:**  
CowCow  
  
  
**Sample Input 2:**  
So  
3  
  
**Sample Output 2:**  
SoSoSo

**import**java.io.\*;

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

**publicclass** Main {

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

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

String s1 = sc.nextLine();

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

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

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

**if** (s1.length() >= 3) {

sb.append(s1.substring(0, 3));

} **else** {

sb.append(s1.substring(0, s1.length()));

}

}

System.*out*.println(sb);

}

}

**106.Regular Expression – 3 (Phone Validator)**

Given a phone number as string, validate the same based on the following rules.  
  
1. Value should contain only numbers.  
2. Value should contain 10 digits.  
3. Value should not start with 00.  
  
If all the conditions are satisifed then print TRUE else print FALSE.  
  
Include a class UserMainCode with a static method **validatePhone** which accepts the string. The return type is the boolean formed based on rules.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a string.  
Output consists of TRUE or FALSE .  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
9987684321  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
0014623452  
  
**Sample Output 2:**  
FALSE

**import**java.io.\*;

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

**publicclass** Main {

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

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

String n = sc.next();

**if** (n.matches("[1-9]{2}[0-9]{8}")) {

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

} **else** {

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

}

}

}  
  
**107.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

**import**java.io.\*;

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

**publicclass** Main {

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

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

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

**int** i = 0, sum = 0;

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

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

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

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

}

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

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

al.add(itr.next());

}

Collections.*sort*(al);

System.*out*.println(al);

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

sum = sum + al.get(i);

}

System.*out*.println(sum);

}

}

**108.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

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

**publicclass** Main {

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

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

String s = sc.nextLine();

**if** (s.matches("[a-zA-Z0-9]{1}[ix]{1}[\\w a-zA-z0-9]{1,}")) {

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

} **else** {

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

}

}

}

**109.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.io.\*;

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

**publicclass** Main {

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

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

**int** sum = 0;

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

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

**if** (n % i == 0) {

sum = sum + i;

}

}

**if** (sum == n) {

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

} **else**

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

}

}

**110.Check Characters in a String**

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

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

The return type of this method is an int.  Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

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

**Input and Output Format:**

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

**Sample Input 1:**

the picture was great

**Sample Output 1:**  
Valid

**Sample Input 1:**

this

**Sample Output 1:**

Invalid

**import**java.io.\*;

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

**publicclass** Main {

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

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

String s = sc.nextLine();

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

sb.reverse();

String s1 = sb.toString();

**if** (s.charAt(0) == s1.charAt(0)) {

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

} **else** {

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

}

}

}

**111.Max Scorer**

Write a program that performs the following actions:

1.     Read n strings as input and stores them as an arraylist. The string consists of student information like name and obtained marks of three subjects. Eg: name-mark1-mark2-mark3 [suresh-70-47-12] The mark would range between 0 – 100 (inclusive).

2.     Write a function **highestScorer**which accepts these the arraylist and returns the name of the student who has scored the max marks. Assume the result will have only one student with max mark.

Include a class UserMainCode with the static method **highestScorer** which accepts the arraylist and returns the name (string) of max scorer.

Create a Class Main which would be used to read n strings into arraylist and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of 1 integer and n strings. The first integer denotes the size of the arraylist, the next n strings are score pattern described above.

Output consists of a string with the name of the top scorer.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

sunil-56-88-23

bindul-88-70-10

john-70-49-65

**Sample Output 1:**

john

**import**java.io.\*;

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

**publicclass** Main {

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

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

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

**int** i;

String k = "", s1 = "";

**int** sum = 0, max = 0;

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

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

al.add(sc.next());

}

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

k = al.get(i);

StringTokenizer st = **new** StringTokenizer(k, "-");

**while** (st.hasMoreTokens()) {

String s = st.nextToken();

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

**int** b = Integer.*parseInt*(st.nextToken());

**int** c = Integer.*parseInt*(st.nextToken());

sum = a + b + c;

**if** (sum > max) {

max = sum;

s1 = s;

}

}

}

System.*out*.println(s1);

}

}

**112.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

**import**java.io.\*;

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

**publicclass** Main {

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

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

String s = sc.next();

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

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

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

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

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

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

} **else**

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

}

}

**113.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 array of employee names who belong to that designation 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 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.io.\*;

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

**publicclass** Main {

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

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

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

**int** i, k = 0;

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

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

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

}

String s = sc.next();

String s1[] = **new** String[n];

**for** (Map.Entry e : hm.entrySet()) {

**if** (e.getValue().equals(s)) {

s1[k] = (String) e.getKey();

k++;

}

}

**for** (i = 0; i < s1.length - 1; i++)

System.*out*.println(s1[i]);

}

}