**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(0)).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=0;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;

}

}

}