**1.**    **Check Sum of Odd Digits**

Write a program to read a number , calculate the sum of odd digits (values) present in the given number.  
  
Include a class **UserMainCode** with a static method **checkSum** which accepts a positive integer . The return type should be 1 if the sum is odd . In case the sum is even return -1 as output.  
  
Create a class **Main** which would get the input as a positive integer and call the static method **checkSum** present in the UserMainCode.  
  
**Input and Output Format:**  
Input consists of a positive integer n.  
Refer sample output for formatting specifications.

**Sample Input 1:**

56895

**Sample Output 1:**

Sum of odd digits is odd.

**Sample Input 2:**

84228

**Sample Output 2:**

Sum of odd digits is even.

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

**public** **class** Main

{

/\*\*

\* **@param** args

\*/

**static** **int** checkodd(**int** n)

{

**int** sum=0,result;

**while**(n!=0)

{

**int** r=n%10;

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

{

sum=sum+r;

}

n=n/10;

}

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

{

result=1;

}

**else**

{

result=-1;

}

**return** result;

}

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

{

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** no=Integer.*parseInt*(br.readLine());

**int** res=Main.*checkodd*(no);

**if**(res==1)

{

System.*out*.println("Sum of odd digits is odd.");

}

**else**

{

System.*out*.println("Sum of odd digits is even.");

}

}

}

**3.**    **Sum of Squares of Even Digits**

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

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

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

**Input and Output Format:**

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

**Sample Input 1:**

56895

**Sample Output 1:**

100

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

**public** **class** Main

{

/\*\*

\* **@param** args

\*/

**static** **int** evenSquare(**int** n)

{

**int** sum=0;

**while**(n!=0)

{

**int** r=n%10;

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

{

sum=sum+(r\*r);

}

n=n/10;

}

**return** sum;

}

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

{

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** no=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*evenSquare*(no));

}

}

**7.**    **Reversing a Number**

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

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

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

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

**Input and Output Format:**

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

**Sample Input 1:**

543

**Sample Output 1:**

345

**Sample Input 1:**

1111

**Sample Output 1:**

1111

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

**public** **class** Main

{

/\*\*

\* **@param** args

\*/

**static** **int** revno(**int** n)

{

**int** sum=0;

**while**(n!=0)

{

**int** r=n%10;

sum=(sum\*10)+r;

n=n/10;

}

**return** sum;

}

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

{

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** no=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*revno*(no));

}

}

**8.**    **Array List Sorting and Merging**

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

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

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

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

**Input and Output Format:**

Input consists of two array lists of size 5.

Output is an array list .

Note - The first element is at index 0.

Refer sample output for formatting specifications.

**Sample Input 1:**

**3**

**1**

**17**

**11**

**19**

**5**

**2**

**7**

**6**

**20**

**Sample Output 1:**

**3**

**11**

**19**

**Sample Input 2:**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**Sample Output 2:**

**3**

**7**

**9**

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

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

**public** **class** Main

{

/\*\*

\* **@param** args

\*/

**public** **static** ArrayList<Integer> sortmerge(ArrayList<Integer> l1, ArrayList<Integer> l2)

{

l1.addAll(l2);

Collections.*sort*(l1);

ArrayList<Integer> l3=**new** ArrayList<Integer>();

l3.add(l1.get(2));

l3.add(l1.get(6));

l3.add(l1.get(8));

**return** l3;

}

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

{

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

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

{

l.add(Integer.*parseInt*(br.readLine()));

}

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

{

l1.add(Integer.*parseInt*(br.readLine()));

}

ArrayList<Integer> al=Main.*sortmerge*(l,l1);

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

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

{

System.*out*.println(it.next());

}

}

}

**14.**  **Sum of Powers of elements in an array**

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

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

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

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

**Input and Output Format:**

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

Output is an integer .

**Sample Input 1:**

4

3

6

2

1

**Sample Output 1:**

12

**Sample Input 2:**

4

5

3

7

2

**Sample Output 2:**

61

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

**class** Main

{

/\*\*

\* **@param** args

\*/

**static** **int** arrayInd(**int** n,**int** s[])

{

**int** sum=0;

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

{

sum=(**int**)(sum+Math.*pow*(s[i],i));

}

**return** sum;

}

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

{

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n=Integer.*parseInt*(br.readLine());

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

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

{

s[i]=Integer.*parseInt*(br.readLine());

}

System.*out*.println(Main.*arrayInd*(n,s));

}

}

**15.Difference between largest and smallest elements in an array**

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

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

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

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

**Input and Output Format:**

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

Size of the array must be >=1

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

**Sample Input 1:**

4

3

6

2

1

**Sample Output 1:**

5

**Sample Input 2:**

4

5

3

7

2

**Sample Output 2:**

5

import java.util.Arrays;

import java.util.Scanner;

public class ArrayDiff

{

/\*\*

\* @param args

\*/

static int getDiff(int n,int a1[])

{

int len=a1.length;

Arrays.sort(a1);

int dif=a1[len-1]-a1[0];

if(len==1)

{

return a1[0];

}

else

return dif;

}

public static void main(String[] args)

{

// TODO Auto-generated method stub

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(ArrayDiff.getDiff(n,a));

}

}

**16.Find the element position in a reversed string array**

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

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

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

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

**Input and Output Format:**

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

Assume the position of first element is 1.

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

**Sample Input 1:**

4

red

green

blue

ivory

ivory

**Sample Output 1:**

2

**Sample Input 2:**

3

grape

mango

apple

apple

**Sample Output 2:**

3

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Main

{

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

System.*out*.println("Enter no of elements");

**int** x=Integer.*parseInt*(br.readLine());

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

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

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

{

a[i]=br.readLine();

}

System.*out*.println("enter an ele to search");

String b=br.readLine();

UserMainCode.*getElementPosition*(a,b);

}

}

**class** UserMainCode

{

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

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

**for**(String n:a)

{

al.add(n);

}

System.*out*.println(al);

Collections.*sort*(al);

System.*out*.println(al);

Collections.*reverse*(al);

System.*out*.println(al);

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

{

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

{

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

}

}

}

}

**17.generate the series**

  Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:

1+3-5+7-9…+/-n.

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

The return type of the output should be an integer .

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

**Input and Output Format:**

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

**Sample Input 1:**

9

**Sample Output 1:**

-3

**Sample Input 2:**

11

**Sample Output 2:**

8

import java.util.\*;

public class Main {

                public static int consecutiveSumSubofOddNos(int n){

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

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

                                                if(i%2!=0)

                                                                l1.add(i);

                                int n1=l1.get(0);

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

                                                if(i%2!=0)

                                                                n1=n1+l1.get(i);

                                                else

                                                                n1=n1-l1.get(i);

                                return n1;

                }

                public static void main(String[] args) {

                                Scanner s=new Scanner(System.in);

                                int n=s.nextInt();

                                System.out.println(consecutiveSumSubofOddNos(n));

                }}

**24.Largest Element**

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

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

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

Assume maximum length of array is 20.

**Input and Output Format:**

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

Output consists of a single Integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

2

3

8

4

5

**Sample Output 1:**

8

**import** java.util.Scanner;

**public** **class** LargestElementInArray

{

/\*\*

\* **@param** args

\*/

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

{

**int** max=0;

**int** len=a.length;

**int** first=a[0];

**int** last=a[len-1];

**int** mid=a[len/2];

**if**(len==1)

{

max=a[0];

}

**if**(first>last && first>mid )

{

max=first;

}

**else** **if**(last>first&& last>mid)

{

max=last;

}

**else**

{

max=mid;

}

**return** max;

}

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

{

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

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(LargestElementInArray.*checkLargestAmongCorner*(a));

}

}

**25.nCr**

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

**Note1 :** n! factorial can be achieved using given formula n!=nx(n-1)x(n-2)x..3x2x1.

**Note2 :** 0! = 1.

Example 5!=5x4x3x2x1=120

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

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

**Input and Output Format:**

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

Output consists of a single Integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

3

**Sample Output 1:**

4

**import** java.util.Scanner;

**public** **class** CalculateNCR

{

/\*\*

\* **@param** args

\*/

**static** **int** getNcr(**int** n,**int** r)

{

**int** factn=1,factr=1,factnr=1;

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

{

factn=factn\*i;

}

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

{

factr=factr\*i;

}

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

{

factnr=factnr\*i;

}

**int** result=(factn)/(factr\*factnr);

**return** result;

}

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

{

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

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

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

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

System.*out*.println(CalculateNCR.*getNcr*(n,r));

}

}

**26.Sum of Common Elements**

Write a program to find out sum of common elements in given two arrays. If no common elements are found print - “No common elements”.

Include a class **UserMainCode** with a static method **getSumOfIntersection** which accepts two integer arrays and their sizes. The return type (integer) should return the sum of common elements.

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

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

Assume the common element appears only once in each array.

**Sample Input 1:**

4

3

2

3

5

1

1

3

9

**Sample Output 1:**

4

**Sample Input 2:**

4

3

2

3

5

1

12

31

9

**Sample Output 2:**

No common elements

import java.util.Scanner;

public class Main {

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int m=sc.nextInt();

int[] a=new int[n];

int[] b=new int[m];

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

 a[i]=sc.nextInt();

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

 b[i]=sc.nextInt();

int u=UserMainCode.display(a,b);

if(u==-1)

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

else

System.out.println(u);}}

public class UserMainCode {

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

{

                                int sum=0;

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

                                {

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

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

                                sum=sum+a[i];

                                }}

                                if(sum==0)

                                return -1;

                                else

                                return sum;

                                }}

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

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

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

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

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

**Input and Output Format:**

Input consists of integer array.

Output is an integer sum.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

2

6

3

4

5

**Sample Output 1:**

208

**import** java.util.Scanner;

**public** **class** SumOfSqCubes

{

/\*\*

\* **@param** args

\*/

**static** **int** sqcubes(**int** a[])

{

**int** len=a.length;

**int** sumeven=0,sumodd=0;

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

{

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

{

sumeven=sumeven+(a[i]\*a[i]);

}

**else**

{

sumodd=sumodd+(a[i]\*a[i]\*a[i]);

}

}

**return** sumeven+sumodd;

}

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

{

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

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( SumOfSqCubes.*sqcubes*(a));

}

}

**47.Unique Number**

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

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

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

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

**Input and Output Format:**

Input consists of an integer .

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

Refer sample output for formatting specifications.

**Sample Input 1:**

123

**Sample Output 1:**

Unique

**Sample Input 2:**

33

**Sample Output 2:**

Not Unique

public class useer{

                public static void main(String[]args)

                {

                                Scanner sc=new Scanner(System.in);

                                int n=sc.nextInt();

                                int []a=new int[100];

                                int i=0,count=0;

                                while(n!=0)

                                {

                                                int num=n%10;

                                                a[i]=num;

                                                i++;

                                                n=n/10;

                                }

                                for(int j=0;j<i-1;j++)

                                {

                                                for(int k=j+1;k<=i-1;k++)

                                                {

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

                                                                                count++;

                                                                }

                                                }}

                                                                if(count>0)

                                                                {

                                                                                System.out.println("Invalid");

                                                                }

                                                                else

                                                                {

                                                                                System.out.println("valid");

                                                                }

                                                                }}

**56.Average of Prime Locations**

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

Round the avarage to two decimal places.

Assume that the array starts with index 0.

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

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

**Input and Output Format:**

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

Output consists of a single Double value.

Refer sample output for formatting specifications.

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

**Sample Input 1:**

8

4

1

7

6

5

8

6

9

**Sample Output 1:**

7.5

import java.util.Scanner;

public class kape{

public static void main(String[] args) {

                Scanner sc=new Scanner(System.in);

int []a=new int[20];

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

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

{

                a[i]=Integer.parseInt(sc.nextLine());

}

System.out.print(kape1.display(n,a));

}}

public class kape1{

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

{

int count=0,sum=0,n1=0;

double avg=0;

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

{

count=0;

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

{

if(i%j==0)

{

count++;

}

}

if(count==2)

{

sum=sum+a[i];

n1=n1+1;

}

}

 avg=(double)(sum)/n1;

return avg;

}}

**57.Common Elements**

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

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

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

Assume that all the elements will be distinct.

**Input and Output Format:**

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

Output consists of a single Integer value.

Refer sample output for formatting specifications.

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

**Sample Input 1:**

4

1

2

3

4

2

3

6

7

**Sample Output 1:**

5

import java.util.Scanner;

public class Main {

public static void main(String [] args){

                Scanner sc=new Scanner(System.in);

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

                int[] a=new int[n];

                int[] b=new int[n];

                for(int i=0;i<n;i++)

                a[i]=Integer.parseInt(sc.nextLine());

                for(int i=0;i<n;i++)

                b[i]=Integer.parseInt(sc.nextLine());

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

}}

public class UserMainCode {

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

{

                                int sum=0;

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

                                {

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

                                {

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

                                sum=sum+a[i];

                                }}

                                if(sum==0)

                                return -1;

                                else

                                return sum;

                }}

**58.Middle of Array**

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

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

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

**Input and Output Format:**

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

Output consists of a single Integer value.

Refer sample output for formatting specifications.

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

**Sample Input 1:**

5

1

5

23

64

9

**Sample Output 1:**

23

import java.util.Scanner;

public class kape {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=sc.nextInt();

int []a=new int[s];

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

{

a[i]=sc.nextInt();

}

System.out.println(kape1.display(a));

}

}

public class kape1

{public static int display(int[] a)

{

int y=a.length/2;

return a[y];

}

**61.Boundary Average**

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

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

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

**Input and Output Format:**

The first line of the input consists of an integer n, that corresponds to the size of the array.

The next n lines consist of integers that correspond to the elements in the array.  
Assume that the maximum number of elements in the array is 10.

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

**Sample Input :**

6

3

6

9

4

2

5

**Sample Output:**

5.5

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;

public class main {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

int size=sc.nextInt();

int n[]=new int[size];

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

{

                n[i]=sc.nextInt();

}

System.out.println(Usermaincode.display(n));

}

}

public class Usermaincode {

    public static float display(int num[])

    {

                   int max = num[0];

                   int min=num[num.length-1];

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

                   {

                       if (num[i] > max)

                          max = num[i];

                       if (num[i]<min)

                                   min=num[i];

                   }

                   return (float)(max+min)/2;

    }

}

**70.Digits**

Write a program to read a non-negative integer n, that returns the count of the occurances of 7 as digit.  
  
Include a class UserMainCode with a static method **countSeven** which accepts the integer value. The return type is integer which is the count value.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a integer.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
717  
  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
4534  
  
**Sample Output 2:**  
0

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

**public** **class** Main

{

**public** **static** **int** display(**int** n)

{

**int** num=0,count=0;

**while**(n!=0)

{

num=n%10;

**if**(num==7)

{

count++;

}

n=n/10;

}

**return** count;

}

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

{

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** no=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*display*(no));

}

}

**73.Digit Comparison**

Write a program to read two integers and return true if they have the same last digit.  
  
Include a class UserMainCode with a static method **compareLastDigit** which accepts two integers and returns boolean. (true / false)  
  
Create a Class Main which would be used to accept two integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of two integer.  
Output consists TRUE / FALSE.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
59  
29  
  
**Sample Output 1:**  
TRUE

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** a=Integer.*parseInt*(br.readLine());

**int** b=Integer.*parseInt*(br.readLine());

System.*out*.println(*characterCheck*(a,b));

}

**public** **static** **boolean** characterCheck(**int** q,**int** w) {

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

**int** c=q%10;

**int** d=w%10;

**if**(c==d)

b=**true**;

**else**

b=**false**;

**return** b;

}

}

**74.Duplicates**

GIven three integers (a,b,c) find the sum. However, if one of the values is the same as another, both the numbers do not count towards the sum and the third number is returned as the sum.  
  
Include a class UserMainCode with a static method **getDistinctSum** which accepts three integers and returns integer.  
  
Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of three integers.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
1  
2  
1  
  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
1  
2  
3  
  
**Sample Output 2:**

6

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** a=Integer.*parseInt*(br.readLine());

**int** b=Integer.*parseInt*(br.readLine());

**int** c=Integer.*parseInt*(br.readLine());

System.*out*.println(*characterCheck*(a,b,c));

}

**public** **static** **int** characterCheck(**int** q,**int** w,**int** e) {

**int** s=0;

**if**(q!=w&&q!=e)

{

s=q+w+e;

}

**else** **if**(q==w&&q==e)

{

s=0;

}

**else** **if**(q==w&&q!=e)

{

s=e;

}

**else** **if**(q!=w&&q==e)

{

s=w;

}

**return** s;

}

}

**79.SumOdd**

Write a program to read an integer and find the sum of all odd numbers from 1 to the given number. [inclusive of the given number]  
  
if N = 9 [ 1,3,5,7,9]. Sum = 25  
  
Include a class UserMainCode with a static method **addOddNumbers** which accepts the number n. The return type is the integer based on the problem statement.  
  
Create a Class Main which would be used to accept the integer and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a integer.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
6  
  
**Sample Output 1:**  
9

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

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

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

**int** a=0;

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

{

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

{

a=a+i;

}

}

**return** a;

}

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

{

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** no=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*sumOf*(no));

}

}

**81.Unique Number**

GIven three integers (a,b,c) , Write a program that returns the number of unique integers among the three.  
Include a class UserMainCode with a static method **calculateUnique** which accepts three integers and returns the count as integer.  
Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of three integers.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
12  
4  
3  
**Sample Output 1:**  
3  
  
**Sample Input 2:**  
4  
-4  
4  
**Sample Output 2:**  
2

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

**public** **class** Main

{

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

{

**int** d=0;

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

{

d=3;

}

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

{

d=1;

}

**else** **if**(a!=b&&a==c)

{

d=2;

}

**else** **if**(a==b&&a!=c)

{

d=2;

}

**return** d;

}

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

{

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** a=Integer.*parseInt*(br.readLine());

**int** b=Integer.*parseInt*(br.readLine());

**int** c=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*display*(a,b,c));

}

}

**83.Scores**

Write a program to read a integer array of scores, if 100 appears at two consecutive locations return true else return false.  
Include a class UserMainCode with a static method **checkScores** which accepts the integer array. The return type is boolean.  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values.  
Output consists of a string that is either 'TRUE' or 'FALSE'.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
1  
100  
100  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
3  
100  
1  
100  
**Sample Output 2:**  
FALSE

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

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

arr[i] = sc.nextInt();

}

System.*out*.println(*consecutiveNum*(arr, n));

}

**public** **static** **boolean** consecutiveNum(**int** arr[], **int** n){

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

**for**(**int** i=0;i<n-1;i++){

**if**(arr[i] == 100){

**if**(arr[i+1] == 100){

b = **true**;

**break**;

}

}

}

**return** b;

}

}

**84.ArrayFront**

Write a program to read a integer array and return true if one of the first 4 elements in the array is 9 else return false.  
Note: The array length may be less than 4.  
Include a class UserMainCode with a static method **scanArray** which accepts the integer array. The return type is true / false.  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values.  
Output consists of TRUE / FALSE.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
6  
1  
2  
3  
4  
5  
6  
**Sample Output 1:**  
FALSE  
  
**Sample Input 2:**  
3  
1  
2  
9  
**Sample Output 2:**  
TRUE

import java.util.Scanner;

public class kapes3 {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=sc.nextInt();

int []a=new int[s];

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

{

a[i]=sc.nextInt();

}

System.out.println(kapes4.display(a));

}

}

 public class kapes4

{public static boolean display(int[] a)

{

int u=0,l=0;

boolean b=false;

if(a.length>=4)

l=4;

else

l=a.length;

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

if(a[i]==9)

u=10;

if(u==10)

b=true;

return b;

}

}

**86.Find Distance**

Write a Program that accepts four int inputs(x1,y1,x2,y2) as the coordinates of two points. Calculate the distance between the two points using the below formula.  
Formula : square root of((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2))  
Then, Round the result to return an int  
Include a class UserMainCode with a static method **findDistance** which accepts four integers. The return type is integer representing the formula.  
Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of four integers.  
Output consists of a single integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
3  
4  
5  
2  
**Sample Output 1:**  
3  
  
**Sample Input 2:**  
3  
1  
5  
2  
**Sample Output 2:**  
2

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

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

**public** **static** **int** display(**int** a,**int** b,**int** c,**int** d) {

**long** q=(**int**)Math.*round*(Math.*sqrt*(((a-c)\*(a-c))+((b-d)\*(b-d))));

**return** (**int**) q;

}

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

{

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** a=Integer.*parseInt*(br.readLine());

**int** b=Integer.*parseInt*(br.readLine());

**int** c=Integer.*parseInt*(br.readLine());

**int** d=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*display*(a, b, c, d));

}

}

**88.Sum of Max & Min**

Write a Program that accepts three integers, and returns the sum of maximum and minimum numbers.  
Include a class UserMainCode with a static method getSumMaxMin which accepts three integers. The return type is integer representing the formula.  
Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of three integers.  
Output consists of a single integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
12  
17  
19  
  
**Sample Output 1:**  
31

public class User

{

                public static int display(int a,int b,int c)

                {

                                int d=0;

                if(a<b&&b<c)

                {

                                d=a+c;

                }

                else if(a<b&&b>c)

                {

                                d=b+c;

                }

                else if(a>b&&b<c)

                {

                                d=a+b;

                }

                return d;

}}

**89.Decimal to Binary Conversion**

Write a Program that accepts a decimal number n, and converts the number to binary.  
  
Include a class UserMainCode with a static method **convertDecimalToBinary** which accepts an integer. The return type is long representing the binary number.  
  
Create a Class Main which would be used to accept the input integer and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of single integer.  
Output consists of a single long.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
5  
**Sample Output 1:**  
101

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n=Integer.*parseInt*(br.readLine());

String s1=Integer.*toBinaryString*(n);

**long** y=Long.*parseLong*(s1);

System.*out*.println(y);

}

}

**95.Largest Difference**

Write a program to read a integer array, find the largest difference between adjacent elements and display the index of largest difference.  
**EXAMPLE:**  
input1: {2,4,5,1,9,3,8}  
output1: 4 (here largest difference 9-1=8 then return index of 9 ie,4)  
Include a class UserMainCode with a static method **checkDifference** which accepts the integer array. The return type is integer.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
2  
4  
5  
1  
9  
3  
8  
**Sample Output 1:**  
4

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

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

**public** **static** **int** getDiffArray(**int**[] n1){

**int** n2,n3=0,n4=0,i;

**for**(i=0;i<n1.length-1;i++){

n2=Math.*abs*(n1[i]-n1[i+1]);

**if**(n2>n3){

n3=n2;

n4=i+1; }}

**return** n4;

}

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n=Integer.*parseInt*(br.readLine());

**int**[] n1=**new** **int**[n];

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

{

n1[i]=Integer.*parseInt*(br.readLine());

}

System.*out*.println(*getDiffArray*(n1));

}

}

**2.Maximum Difference**

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

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

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

**Input and Output Format:**

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

Output consists of an Integer (index).

Refer sample output for formatting specifications.

**Sample Input :**

6

4

8

6

1

9

4

**Sample Output :**

4

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

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

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

**public** **static** **int** getDiffArray(**int**[] n1){

**int** n2,n3=0,n4=0,i;

**for**(i=0;i<n1.length-1;i++){

n2=Math.*abs*(n1[i]-n1[i+1]);

**if**(n2>n3){

n3=n2;

n4=i+1; }}

**return** n4;

}

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n=Integer.*parseInt*(br.readLine());

**int** n1[]=**new** **int**[n];

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

{

n1[i]=Integer.*parseInt*(br.readLine());

}

System.*out*.println(*getDiffArray*(n1));

}

}

**3.Palindrome - In Range**

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

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

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

Note1 : A palindrome number is a number which remains same after reversing its digits.

Note2 : A single digit number is not considered as palindrome.

**Input and Output Format:**

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

Output consists of an Integer (sum of palindromes).

Refer sample output for formatting specifications.

**Sample Input :**

130

150

**Sample Output :**

272

**(131+141 = 272)**

import java.util.\*;

public class Main {

                public static int sumOfPalindromeNos(int n1,int n2){

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

                                for(int i=n1;i<=n2;i++){

                                                int r=0,n3=i;

                                                while(n3!=0){

                                                                r=(r\*10)+(n3%10);

                                                                n3=n3/10;           }

                                                if(r==i)

                                                                l1.add(i);                              }

                                System.out.println(l1);

                                int s=0;

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

                                                s+=l1.get(i);

                                return s;

                }

                public static void main(String[] args) {

                                Scanner s=new Scanner(System.in);

                                System.out.println("enter the range:");

                                int n1=s.nextInt();

                                int n2=s.nextInt();

                                System.out.println("sum of palindrome nos.within given range is:"+sumOfPalindromeNos(n1,n2));

                }

}

**5.Fibonacci Sum**

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

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

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

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

**Input and Output Format:**

Input consists of an integer, which corresponds to n.

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

Refer sample output for formatting specifications.

**Sample Input :**

5

**Sample Output :**

7

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

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

public static int sumOfFibonacci(int n){

int a=0,b=1,c=0,d=1;

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

c=a+b;

a=b; b=c;

d=d+c;

}

return d;

}

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

int n=s.nextInt();

System.out.println(sumOfFibonacci(n));

}

}

**9.Remove 10's**

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

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

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

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

**Sample Input :**

5

1

10

20

10

2

**Sample Output :**

1

20

import java.util.Scanner;

public class main

{

public static void main(String[]args)

{

Scanner sc=new Scanner(System.in);

int size=sc.nextInt();

int[]m=new int[size];

int[]n=new int[size];

int j=0;

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

{

n[i]=sc.nextInt();

}

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

{

if(n[i]!=10)

{

m[j]=n[i];

j++;

}}

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

{

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

}

}

}

**19.Max Admissions**

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

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

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

**Input and Output Format:**

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

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

Refer sample output for formatting specifications.

**Sample Input 1:**

4

2010

200000

2011

300000

2012

45000

2013

25000

**Sample Output 1:**

2011

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**import** java.util.ArrayList;

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

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

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n = Integer.*parseInt*(br.readLine());

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

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

inpList.add(Integer.*parseInt*(br.readLine()));

}

System.*out*.println(UserMainCode.*getYear*(inpList));

}

}

**class** UserMainCode {

**public** **static** Integer getYear(ArrayList<Integer> inpList) {

**int** i,maxyear=0,maxcount=0;

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

{

**int** num=inpList.get(i);

**if**(maxcount<num)

maxcount=num;

}

maxyear=inpList.get((inpList.indexOf(maxcount))-1);

**return** maxyear;

}

}

**20.Sum Non Prime Numbers**

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

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

Example:

input = 9

Prime numbers = 2,3,5 and 7

output = 1+4+6+8+9=28

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

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

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

9

**Sample Output:**

28

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

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

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

**int** sum=0;**int** k=0;**int** sum1=0;

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

{ k=0;

**for**(**int** j=2; j<i; j++)

{

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

k++;

}

**if**(k==0)

{

sum=sum+i;

}

}

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

sum1=sum1+r;

**return** sum1-sum;

}

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

{

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** no=Integer.*parseInt*(br.readLine());

System.*out*.println(Main.*func*(no));

}

}

**24.Add and Reverse**

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

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

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

**Example:**

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

Number = 15

sum = 20 + 25 + 30 + 100 = 175

output = 571

**Input and Output Format:**

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

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

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

Output consists of a single integer.

**Sample Input**

6

10

15

20

25

30

100

15

**Sample Output**

571

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n=Integer.*parseInt*(br.readLine());

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

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

{

a[i]=Integer.*parseInt*(br.readLine());

}

**int** b=Integer.*parseInt*(br.readLine());

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

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

{

**if**(a[i]>b)

{

sum=sum+a[i];

}

}

//System.out.println(sum);

**while**(sum!=0)

{

r=(r\*10)+(sum%10);

sum=sum/10;

}

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

}

}

**26.Sum Squares of Digits**

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

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

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

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

321

**Sample Output:**

14

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** n=Integer.*parseInt*(br.readLine());

*getvalues*(n);

}

**public** **static** **void** getvalues(**int** n) {

**int** a=n;

**int** rem=0;

**int** sum=0;

**while**(a!=0)

{

rem=a%10;

sum=sum+(rem\*rem);

a=a/10;

}

System.*out*.println(sum);

}

}

**27.Even and Odd Index Sum**

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

Example:

input = 23050

evenSum = 2 + 0 + 0 = 2

oddSum = 3 + 5 = 8

output = no

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

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

**Input and Output Format:**

Input consists of an integer.

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

**Sample Input 1:**

23050

**Sample Output 1:**

no

**Sample Input 2:**

231

**Sample Output 2:**

Yes

**import** java.util.Scanner;

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

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

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

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

*getvalues*(n);

}

**public** **static** **void** getvalues(**int** n) {

**int** rem = 0, i = 0;

**int** a[] = **new** **int**[10];

**while** (n > 0) {

rem = n % 10;

a[i] = rem;

n = n / 10;

i++;

}

**int** sume = 0, sumo = 0;

**for** (**int** j = i - 1; j >= 0; j -= 2) {

sumo = sumo + a[j];

}

**for** (**int** j = i - 2; j >= 0; j -= 2) {

sume = sume + a[j];

}

**if** (sume == sumo) {

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

} **else**

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

}

}

**30.Programming Logic**

Write a Program that accepts three integer values (a,b,c) and returns their sum. However, if one of the values is 13 then it does not count towards the sum and the next number also does not count. So for example, if b is 13, then both b and c do not count.  
  
Include a class UserMainCode with a static method **getLuckySum** which accepts three integers. The return type is integer representing the sum.  
  
Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of three integers.  
  
Output consists of a single integer.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
1  
2  
3  
  
**Sample Output 1:**  
6  
  
  
**Sample Input 2:**  
1  
2  
13  
  
**Sample Output 2:**  
3  
  
  
**Sample Input 3:**  
13  
3  
8  
  
**Sample Output 3:**  
8

**import** java.util.ArrayList;

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

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

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

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.*in*));

**int** ip1=Integer.*parseInt*(br.readLine());

**int** ip2=Integer.*parseInt*(br.readLine());

**int** ip3=Integer.*parseInt*(br.readLine());

System.*out*.println(*thirteenLapse*(ip1,ip2,ip3));

}

**public** **static** **int** thirteenLapse(**int** ip1, **int** ip2, **int** ip3) {

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

l.add(ip1);

l.add(ip2);

l.add(ip3);

**int** s=0;

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

**if**(l.get(i)!=13)

s+=l.get(i);

**if**(l.get(i)==13)

i=i+1;}

**return** s;

}}

**31.Triplets**

Given an integer array, Write a program to find if the array has any triplets. A triplet is a value if it appears 3 consecutive times in the array.  
  
Include a class UserMainCode with a static method **checkTripplets** which accepts an integer array. The return type is boolean stating whether its a triplet or not.  
  
Create a Class Main which would be used to accept the input arrayand call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of n+1 integers. The first integer would represent the size of array and the next n integers would have the values.  
  
Output consists of a string stating TRUE or FALSE.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
3  
3  
5  
5  
5  
2  
3  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
7  
5  
3  
5  
1  
5  
2  
3  
  
**Sample Output 2:**  
FALSE

**import** java.util.Scanner;

**public** **class** Triplets

{

/\*\*

\* **@param** args

\*/

**static** **boolean** check(**int** a[])

{

**boolean** result=**false**;

**int** len=a.length;

**for**(**int** i=0;i<len-1;i++)

{

**if**((a[i]==a[i+1]) && (a[i+1]==a[i+2]))

{

result=**true**;

}

}

**return** result;

}

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

{

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

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(Triplets.*check*(a));

}

}

**42.Median Calculation**

Write a program to accept an int array as input, and calculate the median of the same.  
  
Median Calculation Procedure:

1. Sort the sequence of numbers.
2. The total number count is odd, Median will be the middle number.

    The total number count is even, Median will be the average of two middle numbers, After calculating the average, round the number to nearest integer.  
  
Include a class UserMainCode with a static method **calculateMedian** which accepts the int array. The return type is the integer which would be the median.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a an integer which denotes the size of the array followed by the array of integers.  
Output consists of a integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
1  
2  
1  
4  
7  
1  
2  
  
**Sample Output 1:**  
2  
  
**Sample Input 2:**  
6  
52  
51  
81  
84  
60  
88  
  
**Sample Output 2:**  
71

**import** java.util.Arrays;

**import** java.util.Scanner;

**public** **class** MedianCalculation

{

/\*\*

\* **@param** args

\*/

**static** **int** calc(**int** a[])

{

**int** len=a.length;

Arrays.*sort*(a);

**int** mid1,mid2,result=0;

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

{

mid1 = len/2;

mid2 = mid1+1;

**float** add = a[mid1-1]+a[mid2-1];

**float** div = add/2;

result = Math.*round*(div);

}

**else**

{

mid1 = len/2;

result=a[mid1];

}

**return** result;

}

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

{

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

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(MedianCalculation.*calc*(a));

}

}

**43.Sequence in Array**

Write a program to accept an int array as input, and check if [1,2,3] appears somewhere in the same sequence.  
  
Include a class UserMainCode with a static method **searchSequence** which accepts the int array. The return type is a boolean which returns true or false.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a an integer which denotes the size of the array followed by the array of integers.  
Output should print true or false.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
9  
11  
-2  
5  
1  
2  
3  
4  
5  
6  
  
**Sample Output 1:**  
TRUE  
  
**Sample Input 2:**  
6  
-2  
5  
1  
3  
2  
6  
  
**Sample Output 2:**  
FALSE

**import** java.util.Scanner;

**public** **class** ArraySequence

{

/\*\*

\* **@param** args

\*/

**static** **boolean** checkSeq(**int** n,**int** a[])

{

**boolean** result=**false**;

**int** len=a.length;

**for**(**int** i=0;i<len-1;i++)

{

**if**((a[i]==1) && (a[i+1]==2) && (a[i+2]==3))

result=**true**;

}

**return** result;

}

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

{

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

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(ArraySequence.*checkSeq*(n,a));

}

}

**50.Perfect Number**

Write a program to that takes a positive integer and returns true if the number is perfect number.  
  
A positive integer is called a perfect number if the sum of all its factors (excluding the number itself, i.e., proper divisor) is equal to its value.  
  
For example, the number 6 is perfect because its proper divisors are 1, 2, and 3, and 6=1+2+3; but the number 10 is not perfect because its proper divisors are 1, 2, and 5, and 1+2+5 is not equal to 10  
  
Include a class UserMainCode with a static method **getPerfection** which accepts the number. The return type is boolean (true / false).  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a integer.  
Output consists of TRUE / FALSE.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
28  
  
**Sample Output 1:**  
TRUE

**import** java.util.Scanner;

**public** **class** PerfectNumber

{

/\*\*

\* **@param** args

\*/

**static** **boolean** perfect(**int** n)

{

**boolean** res=**false**;

**int** sum=0;

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

{

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

sum=sum+i;

}

**if**(sum==n)

res = **true**;

**else**

res=**false**;

**return** res;

}

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

{

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

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

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

System.*out*.println(PerfectNumber.*perfect*(n));

}

}

**64.Largest Span**

Write a program to read a integer array, find the largest span in the array.  
Span is the count of all the elements between two repeating elements including the repeated elements.  
Include a class UserMainCode with a static method **getLargestSpan** which accepts the integer array. The return type is integer.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer n which is the number of elements followed by n integer values.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
6  
4  
2  
1  
4  
5  
7  
**Sample Output 1:**  
4

import java.util.Scanner;

public class Main {

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

 int n=sc.nextInt();

 int []a=new int[n];

 for(int i=0;i<n;i++)

 {

                 a[i]=sc.nextInt();

 }

 System.out.print(UserMainCode.display(a,n));

}}

public class UserMainCode {

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

                                {

                                                int gap=0,max=0;

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

                                                {

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

                                                                {

                                                                                if(x[i]==x[j])

                                                                                {

                                                                                                gap=j;

                                                                                }

                                                                }

                                                                if(gap-i>max)

                                                                                max=gap-i;

                                                }

                                                return max+1;

                                }

                }

**15.Kaprekar Number**

Write a program to check whether the given input number is a Kaprekar number or not.

**Note :**A positive whole number ‘n’ that has ‘d’ number of digits is squared and split into two pieces, a right-hand piece that has ‘d’ digits and a left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of the two pieces is equal to the number, then ‘n’ is a Kaprekar number.

If its Kaprekar number assign to output variable 1 else -1.

Example 1:

Input1:9

9^2 = 81, right-hand piece of 81 = 1 and left hand piece of 81 = 8

Sum = 1 + 8 = 9, i.e. equal to the number. Hence, 9 is a Kaprekar number.

Example 2:

Input1:45

Hint:

45^2 = 2025, right-hand piece of 2025 = 25 and left hand piece of 2025 = 20

Sum = 25 + 20 = 45, i.e. equal to the number. Hence, 45 is a Kaprekar number."

Include a class **UserMainCode** with a static method “**getKaprekarNumber**” that accepts an integer argument and returns an integer. The method returns 1 if the input integer is a Kaprekar number. Else the method returns -1.

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

**Input and Output Format:**

Input consists of an integer.

Output consists of a single string that is either “Kaprekar Number” or “Not A Kaprekar Number”

**Sample Input 1:**

9

**Sample Output 1:**

Kaprekar Number

**Sample Input 2:**

45

**Sample Output 2:**

Kaprekar Number

**Sample Input 3:**

4

**Sample Output 3:**

Not A Kaprekar Number

**import** java.util.Scanner;

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

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

{

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

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

useerm.*display*(n);

}}

**class** useerm{

**public** **static** **void** display(**int** a)

{

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

**int** a1=a;

**while**(a1!=0)

{

count=count+1;

a1=a1/10;

}

**int** square=a\*a;

String s=Integer.*toString*(square);

String s1=s.substring(0,count);

String s2=s.substring(count);

**int** x=Integer.*parseInt*(s1);

**int** y=Integer.*parseInt*(s2);

**int** result =x+y;

**if**(result==a){

j=1;

}

**else**

{

j=2;

}

**if**(j==1)

{

System.*out*.println("Kaprekar Number");

}

**else**

System.*out*.println("not an Kaprekar Number");

}}

**18.average of primes**

Write a program to read an array and find average of all elements located at index i, where i is a prime number. Type cast the average to an int and return as output. The index starts from 0.

Include a class UserMainCode with a static method **addPrimeIndex** which accepts a single integer array. The return type (integer) should be the average of all elements located at index i where i is a prime number.

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

**Input and Output Format:**

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

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20 and minimum number of elements is 3.

**Sample Input 1:**

4

2

5

2

4

**Sample Output 1:**

3

**import** java.util.Scanner;

**public** **class** AvgPrimeIndex

{

/\*\*

\* **@param** args

\*/

**static** **int** getvalues(**int** n,**int** a[])

{

**int** sum=0,sum\_count=0;

**int** len=a.length;

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

{

**int** count=0;

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

{

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

{

count++;

}

}

**if**(count==2)

{

sum=sum+a[i];

sum\_count++;

}

}

**int** avg=sum/sum\_count;

**return** avg;

}

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

{

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

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(AvgPrimeIndex.*getvalues*(n,a));

}

}

**25.Sum of Digits**

Write a Program that accepts a word as a parameter, extracts the digits within the string and returns its sum.

Include a class UserMainCode with a static method **getdigits** which accepts a string. The return type is integer representing the sum.

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

**Input and Output Format:**

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

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

abc12de4

**Sample Output 1:**

7

**import** java.util.Scanner;

**public** **class** SumOfDigits

{

/\*\*

\* **@param** args

\*/

**static** **int** sum(String s1)

{

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

**int** sum=0;

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

{

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

**if**(Character.*isDigit*(c))

{

String ss=c+"";

sum=sum+Integer.*parseInt*(ss);

}

}

**return** sum;

}

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

{

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

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

String s=sc.next();

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

}

}

**29.Digits - II**

Write a program to read a non-negative integer n, compute the sum of its digits. If sum is greater than 9 repeat the process and calculate the sum once again until the final sum comes to single digit.Return the single digit.  
Include a class UserMainCode with a static method **getDigitSum** which accepts the integer value. The return type is integer.  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of a integer.  
Output consists of integer.  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
9999  
**Sample Output 1:**  
9  
  
**Sample Input 2:**  
698  
**Sample Output 2:**  
5

**import** java.util.Scanner;

**public** **class** Digits

{

/\*\*

\* **@param** args

\*/

**static** **int** sum(**int** n)

{

**int** sum = 0 ;

//int n1=n;

**while**(n>10)

{

**int** r = 0 ;

sum = 0;

**while**(n!=0)

{

r = n%10;

sum=sum+r;

n=n/10;

}

n=sum;

}

**return** sum;

}

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

{

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

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

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

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

}

}

**31.Shift Left**

Write a program to read a integer array of scores, and return a version of the given array where all the 5's have been removed. The remaining elements should shift left towards the start of the array as needed,  
  
and the empty spaces at the end of the array should be filled with 0.  
  
So {1, 5, 5, 2} yields {1, 2, 0, 0}.  
  
Include a class UserMainCode with a static method shiftLeft which accepts the integer array. The return type is modified array.  
  
Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of an integer n which is the number of elements followed by n integer values.  
  
Output consists of modified array.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
7  
1  
5  
2  
4  
5  
3  
5  
  
**Sample Output 1:**  
1  
2  
4  
3  
0  
0  
0

**import** java.util.Scanner;

**public** **class** ShiftLeft

{

/\*\*

\* **@param** args

\*/

**static** **int**[] shift(**int** n,**int** a[])

{

**int** ans[]=**new** **int**[a.length];

**int** k=0;

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

{

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

{

ans[k]=a[i];

k++;

}

}

**return** ans;

}

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

{

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

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

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

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

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

{

a[i]=sc.nextInt();

}

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

res=ShiftLeft.*shift*(n,a);

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

{

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

}

}

}

**35.Sequence Sum**

Write a program to read a non-negative integer n, and find sum of fibonanci series for n number..  
  
Include a class UserMainCode with a static method **getFibonacciSum** which accepts the integer value. The return type is integer.  
  
The fibonacci seqence is a famous bit of mathematics, and it happens to have a recursive definition.  
  
The first two values in the sequnce are 0 and 1.  
  
Each subsequent value is the sum of the previous two values, so the whole seqence is 0,1,1,2,3,5 and so on.  
  
You will have to find the sum of the numbers of the Fibonaaci series for a given int n.  
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
  
Input consists of a integer.  
  
Output consists of integer.  
  
Refer sample output for formatting specifications.  
  
**Sample Input 1:**  
  
5  
  
**Sample Output 1:**  
  
7

**import** java.util.Scanner;

**public** **class** SeqSum

{

/\*\*

\* **@param** args

\*/

**static** **int** sumOfFib(**int** n)

{

**int** first = 0, second = 1, next,sum=0;;

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

{

**if** ( i <= 1 )

next = i;

**else**

{

next = first + second;

first = second;

second = next;

}

sum=sum+next;

}

**return** sum;

}

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

{

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

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

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

System.*out*.println(*sumOfFib*(n));

}

}