Q1. Take an input n which is size of array then n more inputs the values in array then find the square of special elements.
An element nums[i] of nums is called special if i divides n, i.e. n % i == 0.
Return *the sum of the squares of all special elements of* nums.

Example 1:
Input:
4
1 2 3 4

Output: 21

Explanation: There are exactly 3 special elements in nums: nums[1] since 1 divides 4, nums[2] since 2 divides 4, and nums[4] since 4 divides 4.
Hence, the sum of the squares of all special elements of nums is nums[1] * nums[1] + nums[2] * nums[2] + nums[4] * nums[4] = 1 * 1 + 2 * 2 + 4 * 4 = 21.

Example 2:
Input:
6
2 7 1 19 18 3

Output: 63

Explanation: There are exactly 4 special elements in nums: nums[1] since 1 divides 6, nums[2] since 2 divides 6, nums[3] since 3 divides 6, and nums[6] since 6 divides 6.
Hence, the sum of the squares of all special elements of nums is nums[1] * nums[1] + nums[2] * nums[2] + nums[3] * nums[3] + nums[6] * nums[6] = 2 * 2 + 7 * 7 + 1 * 1 + 3 * 3 = 63.
Constraints:

- 1 <= nums.length == n <= 50
- 1 <= nums[i] <= 50

Solution:

```
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(sumOfSquares(arr));
      }
    public static int sumOfSquares(int[] nums) {
    int ans = 0;
   for (int i = 0; i < nums.length; i++) {
   if (nums.length % (i + 1) == 0)
    ans += (nums[i] * nums[i]);
}
return ans;
}
}
```

Q2)Print Characters

Amy is a high school student who is passionate about coding. One day, her computer science teacher gives the class an assignment to print all the characters of a given string in separate lines.

Amy immediately gets to work and writes a simple program. However, she feels that her solution is too basic and wants to find a more efficient way to solve the problem.

Can you help Amy by writing a program that prints all the characters of a given string in separate lines.

Input Format

Input contains a String. Constraints 1<=str.length<=10000

Output Format

Print each character in different line

Sample Input 0

String

Sample Output 0

S

t

r

i

n

g

Test cases:-

Input	Output
Abcd	a b c d
CODING	C O D I N G
Car	C a r
String	S t r i n g
Question	Q u e s t i o n

Solution:-

```
import java.util.Scanner;
public class P1
{
public static void main(String[] args)
   {
   Scanner sc=new Scanner(System.in);
   String str=sc.next(); print(str);
   }
public static void print(String str)
{   for(int i=0;i<str.length();i++)
   {
    System.out.println(str.charAt(i));
   }
}
}
```

Q3)Is It Equal?

Once there was a girl named Sarah who loved to write poetry. She had a habit of writing down her thoughts in a notebook whenever she felt inspired. One day, while she was working on a new piece, she accidentally spilled her coffee on the notebook.

Desperate to salvage her work, she decided to copy the poem onto a new page.

However, when she finished rewriting it, she noticed that there were a few discrepancies between the original version and the new one. She wondered if she had missed anything while transcribing the poem.

Help Sarah and write a program that checks if two strings are identical or not.

Input Format

First line contains string s1.

Second line contains string s2.

Constraints

1 <= string1.length() <= 100000

1 <= string2.length() <= 100000

Output Format

Return A boolean value

Sample Input 0

COLLEGE

COLLEGE

Sample Output 0

true

Explanation 0

since both strings have the same character at each index. therefore, it is an equal string.

Input	Output
coding coding	true
String String	true
abcder abcdef	false
Development Development	true
placement placement	false

Solution:-

```
import java.util.Scanner;

public class P2

{

public static void main(String[] args)

   {

     Scanner sc=new Scanner(System.in);

     String str1=sc.next();

     String str2=sc.next();

     System.out.println(equal(str1,str2));

    }
```

```
  public static boolean equal(String str1,String str2)
  {
    if(str1.length()!=str2.length())
    {
     return false;
    }
    else
    {
     for(int i=0;i<str1.length();i++)
     {
      if(str1.charAt(i)!=str2.charAt(i))
      {
       return false;
      }
     }
    }
  return true;
}
}
```

Q4) Palindromic String

You have been given a String S. You need to find and print whether this string is a palindrome or not. If yes, print "YES" (without quotes), else print "NO" (without quotes). Input Format

The first and only line of input contains the String S. The String shall consist of lowercase English alphabets only.

Output Format

Print the required answer on a single line.

Constraints

$1 \leq |S| \leq 100$

Note:-String S consists of lowercase English Alphabets only.

Sample Input

aba

Sample Output

YES

est Cases:-

Input	Output
Abababababa	YES
Racecar	YES
Hellolleh	YES
Acdcdcdaba	NO
Cricketer	NO

Solution:-

```
import java.util.Scanner; public class P3 {
public static boolean isPalindrome(String str)
{
int i = 0, j = str.length() - 1; while (i < j) {
if (str.charAt(i) != str.charAt(j)) return false;
i++; j--;
}
return true;
}
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in); String str=sc.next();
if (isPalindrome(str))
// It is a palindrome System.out.print("YES");
else
// Not a palindrome System.out.print("NO");
}}
```

Q5)Print Indices of Vowels

Maggie is a language enthusiast who loves exploring the intricacies of different languages. One day, while studying English, she comes across a coding challenge that involves printing the indices of vowels in a given string.

Maggie is determined to solve the challenge and begins working on the problem.

Help Maggie and write a program that prompts the user to input a string, and then scans the string for vowels while keeping track of the indices. Whenever you find a vowel, print the index.

Input Format

Input contains a String str.

Constraints

1 <= str.length() <= 10^4

Output Format

Return An series of integer numbers in a single line.

Sample Input 0

aqua

Sample Output 0

0 2 3

Explanation 0

at index 0 we have a

at index 2 we have

u at index 3 we have a

Input	Output
Hello	1 4
University	0 2 4 7
Water	1 3
Programming	2 5 8
Section	1 4 5

Solution:-

```
import java.util.Scanner;
public class P4 {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String str1=sc.next();
print(str1);
}
public static void print(String str){
for(int i=0;i<str.length();i++){
char ch =str.charAt(i);
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'){
System.out.print(i+" ");
}
}
}
}
```

Q6)Toggle String

You have been given a String S consisting of uppercase and lowercase English alphabets. You need to change the case of each alphabet in this String. That is, all the uppercase letters should be converted to lowercase and all the lowercase letters should be converted to uppercase. You need to then print the resultant String to output.

Input Format

The first and only line of input contains the String S

Output Format

Print the resultant String on a single line.

Constraints

$1 \leq |S| \leq 100$

where S denotes the length of string S.

Sample Input:-abcdE

Sample Output:-ABCDe

Test cases:-

Input	Output
String	STRIng
bcdEFrTuv	BCDefRtUV
Hello	hELLO
Gla	GLA
AppLE	aPPle

Solution:-

```
import java.util.Scanner;
public class P5 {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String str=sc.next();
toggle(str);
}
public static void toggle(String s){
String s1 = "";
for (int i = 0; i < s.length(); i++) {
if(Character.isUpperCase(s.charAt(i)))
{
s1=s1+Character.toLowerCase(s.charAt(i));
}
else
{
s1=s1+Character.toUpperCase(s.charAt(i));
}
}
```

```
System.out.println(s1);
}
}
```

Q7)Search Character

Given a small case character ch and an array containing only the small case alphabets, you have to print the index if the character ch is present in the array. If no such character found print -1.

Input Format

- An Character ch
- An integer value representing size of array
- n character value representing elements of array.

Constraints

- 'a'<=ch<='z'
- 1<=n<=100000
- 'a'<=arr[i]<='z'

Output Format

A Character value

Sample Input 0

```
c 5
a b c d e
```

Sample Output 0

```
2
```

Explanation 0

since d is just greater than the c in the given array.

Test Case:-

Input	Output
d 6 a b c f d e	4
e 10	-1

a p p l b a n a n a	
z 5 z e b r a	0
r 6 e f g h r j	4

Solution:-

```
import java.util.Scanner;

public class P7 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

String inputString = scanner.nextLine();

char targetChar = scanner.next().charAt(0);

int index = searchCharacter(inputString, targetChar);

System.out.println(index);

}

public static int searchCharacter(String inputString, char targetChar)
{

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

if (inputString.charAt(i) == targetChar) {

return i;

}

}

return -1;

}

}
```

Q9)Count Words

Samantha was a college student majoring in English literature. One day, her professor assigned the class a writing exercise where they had to write a short story. Samantha was

excited about the challenge, but she wasn't sure how to keep track of the word count as she wrote.

Can you create a program that can count the number of words present in Samantha's short story?

Input Format

Input contains a String str.

Constraints

1<=Str.length()<=100000

Output Format

Return An integer value.

Sample Input 0

Welcome to GLA

Sample Output 0

3

Explanation 0

3 words are present in string i.e welcome, to, GLA.

Test Case:-

Input	Output
Welcome to Gla university	4
I am btech student	4
Hello coder	2
Dsa is necessary for placement	5
The quick brown fox jumps over the lazy dogs	9

Solution:-

```
import java.util.Scanner;

public class P9 {

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

```
Scanner sc=new Scanner(System.in);
String samanthaStory=sc.nextLine();
int numWords = countWords(samanthaStory);
System.out.println(numWords);
}
public static int countWords(String text) {
// Split the text into words using whitespace as the delimiter
String[] words = text.split("\\s+");
// Count the number of words return words.length;
}
}
```

Q10)Count Vowel

In the magical kingdom of Lingua, where words held immense power, there lived a young linguist named Ava. One day, Ava embarked on a quest to count the vowels in a mysterious sentence said to unlock hidden treasures. The sentence was inscribed on an ancient scroll, and its vowels were said to be the key to finding the kingdom's greatest secret. Can you help Ava decipher the sentence and uncover the hidden riches of Lingua?

Input Format

Input contains a Sentence str.

Constraints

Sentence str containing only the small case alphabet.

Sample Input 0

hello how are you

Expected Output 0

7

Explanation:

In this test case, the input sentence contains 7 vowels ('e', 'o', 'o', 'a', 'e', 'o', 'u').

Test cases:-

Input	Output
rhythm and blues	3
the quick brown fox jumps over the lazy dog	11
welcome to gla	5
welcome to coding world	7
coding is essentials for placements	11

Solution:-

```
import java.util.Scanner;
public class P8 {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String str1=sc.next();
System.out.println(cnt(str1));
}
public static int cnt(String str){
int count=0;
for(int i=0;i<str.length();i++){
char ch =str.charAt(i);
if(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u'){
count++;
}
}
return count;
}
```

}

Q11)Strings-Remove Duplicates

Take as input S, a string. Write a function that removes all consecutive duplicates. Print the value returned.

Input Format

String

Constraints

A string of length between 1 to 1000

Output Format

String

Sample Input

aabccba

Sample Output

abcba

Explanation

For the given example, "aabccba", Consecutive Occurrence of a is 2, b is 1, and c is 2. After removing all of the consecutive occurrences, the Final ans will be : - "abcba".

Test Case:-

Input	Output
aabbcc	Abc
aabbccdd	Abcd
abccdeefghii	abcdefghi
aapplleerrtt	aplert
helloppinnhh	helopinh

Solution:-

```
import java.util.Scanner;
public class RemoveConsecutiveDuplicates {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String inputString=sc.nextLine();
String result = removeConsecutiveDuplicates(inputString);
System.out.println(result);
}
public static String removeConsecutiveDuplicates(String
inputString) {
if (inputString == null || inputString.length() == 0) {
return inputString;
}
StringBuilder result = new StringBuilder();
char prevChar = inputString.charAt(0);
result.append(prevChar);
for (int i = 1; i < inputString.length(); i++) {
char currentChar = inputString.charAt(i);
if (currentChar != prevChar) {
result.append(currentChar);
prevChar = currentChar;
}
}
return result.toString();
}
}
}
```

12.One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed

w kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.

Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

Input Format

The first (and the only) input line contains integer number w – the weight of the watermelon bought by the boys.

Constraints

(1 ≤ w ≤ 100)

Output Format

Print YES, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and NO in the opposite case.

Sample Input 0

8

Sample Output 0

YES

Explanation 0

For example, the boys can divide the watermelon into two parts of 2 and 6 kilos respectively (another variant – two parts of 4 and 4 kilos).

Test case

1. 8 Output :- YES
2. 2 Output:- NO
3. 10 Output :- YES
4. 19 Output:- NO
5. 20 Output:- YES

Solution

```
import java.io.*;
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        Scanner sc= new Scanner(System.in);
        int w=sc.nextInt();
        if(w!=2 && w%2==0)
            {
                System.out.println("YES");
            }
             else
              {
            System.out.println("NO");
              }
    }
}
```

13.Capitalization is writing a word with its first letter as a capital letter. Your task is to capitalize the given word.

Note, that during capitalization all the letters except the first one remains unchanged.

Input Format

A single line contains a non-empty word. This word consists of lowercase and uppercase English letters.

Constraints

- The length of the word will not exceed 103.

Output Format

Output the given word after capitalization.

Sample Input 0

ApPLe

Sample Output 0

ApPLe

Sample Input 1

konjac

Sample Output 1

Konjac

Test case

1. ApPLe Output :- ApPLe
2. Konjac Output:- Konjac
3. gLA Output :- GLA
4. WELCOME Output:- WELCOME
5. university Output:- University

Solution

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc =new Scanner(System.in);
            String s=sc.next();
        if (s.charAt(0)>=97 && s.charAt(0)<=122)
        {
```

```
            s=(char)(s.charAt(0)-32)+s.substring(1);
            System.out.println(s);
        }
        else {
        System.out.println(s);
    }
    }
}
```

14.Petya loves football very much. One day, as he was watching a football match, he was writing the players' current positions on a piece of paper. To simplify the situation he depicted it as a string consisting of zeroes and ones. A zero corresponds to players of one team; a one corresponds to players of another team. If there are at least 7 players of some team standing one after another, then the situation is considered dangerous. For example, the situation 00100110111111101 is dangerous and 11110111011101 is not. You are given the current situation. Determine whether it is dangerous or not.

Input Format

The first input line contains a non-empty string s consisting of characters "0" and "1", which represents players. The length of the string does not exceed 100 characters. There's at least one player from each team present on the field.

Constraints

1 < |s| <= 100

Output Format

Print YES if situation is dangerous, else NO

Sample Input 0

001001

Sample Output 0

NO

Test case

1. 001001 Output :- NO
2. 1100000001111111 Output:- YES
3. 00000011111100000011 Output :- NO
4. 11001111111111000 Output:- YES
5. 10101010101010101 Output:- NO

Solution

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner sc =new Scanner(System.in);
        String s=sc.nextLine();
        if(s.contains("1111111")||s.contains("0000000")){
            System.out.println("YES");
        }
        else{
            System.out.println("NO");
        }
    }
}
```

15.Take N as input. Print the sum of its odd placed digits and sum of its even placed digits.

Input Format

Constraints

0 < N <= 1000000000

Output Format

Sample Input

2635

Sample Output

11

5

Explanation

5 is present at 1st position, 3 is present at 2nd position, 6 is present at 3rd position and 2 is present at 4th position.

Sum of odd placed digits on first line. 5 and 6 are placed at odd position. Hence odd place sum is 5+6=11

Sum of even placed digits on second line. 3 and 2 are placed at even position. Hence even place sum is 3+2=5

Solution

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

    public static void main(String args[]) {
Scanner sc=new Scanner(System.in);
     String s[]=sc.next().split("");
     int a[]=new int[s.length];
     for(int i=0;i<s.length;i++)
     {
           a[i]=Integer.parseInt(s[i]);
     }
     int even=0,odd=0;
     for(int i=a.length-1;i>=0;i--)
     {
           int pos=a.length-i+1;
           if(pos%2==0)
```

```
            even=even+a[i];
            else
            odd=odd+a[i];
      }
      System.out.println(even);
      System.out.println(odd);

     }
}
```

16.Take the following as input.

A number

A digit

Write a function that returns the number of times digit is found in the number. Print the value returned.

Input Format

Integer (A number) Integer (A digit)

Constraints

0 <= N <= 1000000000 0 <= Digit <= 9

Output Format

Integer (count of times digit occurs in the number)

Sample Input

5433231

3

Sample Output

3

Explanation

The digit can be from 0 to 9. Assume decimal numbers. In the given case digit 3 is occurring 3 times in the given number.

Solution

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

    public static void main(String args[]) {
Scanner sc=new Scanner(System.in);
     String s[]=sc.next().split("");
     int a[]=new int[s.length];
     for(int i=0;i<s.length;i++)
     {
          a[i]=Integer.parseInt(s[i]);
     }
int c=0;
     int k=sc.nextInt();
     for(int i:a)
     {
          if(i==k)
          c++;
     }
     System.out.println(c);
     }
}
```

17.Take N as input, Calculate it's reverse also Print the reverse.

Input Format

Constraints

0 <= N <= 1000000000

Output Format

Sample Input

123456789

Sample Output

987654321

Explanation

You've to calculate the reverse in a number, not just print the reverse.

Solution

```
import java.util.*;

public class Main {


    public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

     String s[]=sc.next().split("");

     int a[]=new int[s.length];

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

     {

           a[i]=Integer.parseInt(s[i]);

     }

for(int i=a.length-1;i>=0;i--)

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

     }

}
```

18.Take the following as input.

Minimum Fahrenheit value

Maximum Fahrenheit value

Step

Print as output the Celsius conversions. Use the formula C = (5/9)(F – 32) E.g. for an input of 0, 100 and 20 the output is

0 -17

20 -6

40 4

60 15

80 26

100 37

Input Format

The first line of the input contains an integer denoting the Minimum Fahrenheit value. The second line of the input contains an integer denoting the Maximum Fahrenheit value. The third line of the input contains an integer denoting the Step.

Constraints

0 < Min < 100

Min < Max < 500

0 < Step<150

Output Format

Print Fahrenheit and Celsius values separated by a tab. Each step should be printed in a new line.

Sample Input

0

100

20

Sample Output

0 -17

20 -6

40 4

60 15

80 26

100 37

Explanation

First number in every output line is fahrenheit, second number is celsius. The two numbers are separated by a tab.

Solution

```
import java.util.*;

public class Main {

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

```
   Scanner sc=new Scanner(System.in);
         int min=sc.nextInt();
           int max=sc.nextInt();
           int step=sc.nextInt();
           int c;
           for(int i=min;i<=max;i=i+step)
           {
                 c=(int)((5/9.0)*(i-32));
                 System.out.println(i+" "+c);
           }
     }
}
```

19.Take the following as input.

A number

Assume that for a number of n digits, the value of each digit is from 1 to n and is unique. E.g. 32145 is a valid input number.

Write a function that returns its inverse, where inverse is defined as follows

Inverse of 32145 is 12543. In 32145, “5” is at 1st place, therefore in 12543, “1” is at 5th place; in 32145, “4” is at 2nd place, therefore in 12543, “2” is at 4th place.

Print the value returned.

Input Format

Integer

Constraints

0 < N < 1000000000

Output Format

Integer

Sample Input

32145

Sample Output

12543

Explanation

Assume that for a number of n digits, the value of each digit is from 1 to n and is unique. E.g. 32145 is a valid input number. Inverse of 32145 is 12543. In 32145, “5” is at 1st place, therefore in 12543, “1” is at 5th place; in 32145, “4” is at 2nd place, therefore in 12543, “2” is at 4th place

9.

Take as input a number N, print "Prime" if it is prime if not Print "Not Prime".

Input Format

Constraints

2 < N <= 1000000000

Output Format

Sample Input

3

Sample Output

Prime

Explanation

The output is case specific

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int f=0;
        if(a==2 || a==3)
        f=0;
        for(int i=2;i<=a/2;i++)
```

```
        {
            if(a%i==0)
            {
                f=1;
                break;
            }
        }
        if(f==0)
        System.out.print("Prime");
        else
        System.out.print("Not Prime");

    }
}
```

20.Given a integer as a input and replace all the '0' with '5' in the integer

Input Format

Enter an integer n

Constraints

0<=n<=1000000000000

Output Format

All zeroes are replaced with 5

Sample Input

102

Sample Output

152

Explanation

Check each digit , if it is nonzero, then no change required but if it is zero then replace it by 5.

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        String s[]=sc.next().split("");
        int a[]=new int[s.length];
        for(int i=0;i<s.length;i++)
        a[i]=Integer.parseInt(s[i]);
            for(int i=0;i<a.length;i++)
            {
               if(a[i]==0)
               a[i]=5;
            }
    for(int i=0;i<a.length;i++)
    System.out.print(a[i]);
    }
}
```

21.Take the following as input.

A number (N1)

A number (N2)

Write a function which prints first N1 terms of the series 3n + 2 which are not multiples of N2.

Input Format

Constraints

0 < N1 < 100 0 < N2 < 100

Output Format

Sample Input

10

4

Sample Output

5

11

14

17

23

26

29

35

38

41

Explanation

The output will've N1 terms which are not divisible by N2.

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
    int n1=sc.nextInt();
    int n2=sc.nextInt();
    int c=0;
    for(int i=1;c<n1;i++)
    {
        int k=3*i+2;
        if(k%n2==0)
        continue;
        else
        {
            System.out.println(k);
            c++;
```

```
        }
    }
    }
}
```

22.Take the following as input.

A number (N1)

A number (N2)

Write a function which prints all Armstrong numbers between N1 and N2 (inclusive).

371 is an Armstrong number as 371 = 3^3 + 7^3 + 1^3

Input Format

Constraints

0 < N1 < 100 N1 < N2 < 10000

Output Format

Sample Input

400

1000

Sample Output

407

Explanation

Each number in output is in separate line.

Sol

```
import java.util.*;
import java.lang.Math;
public class Main {
    static int get(int num)
    {
        int c=0;
        int num2,num3,sum=0;
```

```
        num2=num;
        num3=num;
        while(num2>0)
        {
            num2=num2/10;
            c++;
        }
        while(num>0)
        {
            int a=num%10;
            sum=sum+(int)(Math.pow(a,c));
            num=num/10;
        }
        if(num3==sum)
        return 1;
        else
        return 0;
    }
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
    int n1=sc.nextInt();
        int n2=sc.nextInt();
        for(int i=n1;i<=n2;i++)
        {
            int k=get(i);
            if(k==1)
            System.out.println(i);
        }

    }
```

}

23.Due to an immense rise in Pollution, Delhi Government is back with the Odd and Even Rule in Delhi. The scheme is as follows, each car will be allowed to run on Sunday if the sum of digits which are even is divisible by 4 or sum of digits which are odd in that number is divisible by 3. However to check every car for the above criteria can't be done by the Delhi Police. You need to help Delhi Police by finding out if a car numbered N will be allowed to run on Sunday?

Input Format

The first line contains N , then N integers follow each denoting the number of the car.

Constraints

N<=1000 Car No >=0 && Car No <=1000000000

Output Format

N lines each denoting "Yes" or "No" depending upon whether that car will be allowed on Sunday or Not !

Sample Input

2

12345

12134

Sample Output

Yes

No

Explanation

1 + 3 + 5 = 9 which is divisible by 3

1 + 1 + 3 = 5 which is NOT divisible by 3 and 2+4 = 6 which is not divisble by 4.

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online
import java.util.*;
class Main {
    public static void main(String[] args) {
```

```
        Scanner sc=new Scanner(System.in);
        int t=sc.nextInt();
        while(t>0)
        {
            int a=sc.nextInt();
            int even=0,odd=0,r;
            while(a>0)
            {
                r=a%10;
                if(r%2==0)
                even=even+r;
                else
                odd=odd+r;
                a=a/10;

            }
            if(even%4==0 || odd%3==0)
            System.out.println("Yes");
            else
               System.out.println("No");
               t--;

        }

           }
}
```

24.Take the following as input.

A number

Write a function which returns true if the number is an Armstrong number and false otherwise, where Armstrong number is defined as follows.

A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

abcd… = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ….

1634 is an Armstrong number as 1634 = 1^4 + 6^4 + 3^4 + 4^4

371 is an Armstrong number as 371 = 3^3 + 7^3 + 1^3

Input Format

Single line input containing an integer

Constraints

0 < N < 1000000000

Output Format

Print boolean output for each testcase.

"true" if the given number is an Armstrong Number, else print "false".

Sample Input

371

Sample Output

true

Explanation

Use functions. Write a function to get check if the number is armstrong number or not. Numbers are armstrong if it is equal to sum of each digit raised to the power of number of digits.

Solution

```
import java.util.*;
import java.lang.Math;
public class Main {
    static int get(int num)
    {
        int c=0;
```

```
        int num2,num3,sum=0;
        num2=num;
        num3=num;
        while(num2>0)
        {
            num2=num2/10;
            c++;
        }
        while(num>0)
        {
            int a=num%10;
            sum=sum+(int)(Math.pow(a,c));
            num=num/10;
        }
        if(num3==sum)
        return 1;
        else
        return 0;
    }
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
    int n1=sc.nextInt();

            int k=get(n1);
            if(k==1)
            System.out.println("true");
            else
             System.out.println("false");
    }
}
```

25.Given a list of numbers, stop processing input after the cumulative sum of all the input becomes negative.

Input Format

A list of integers to be processed

Constraints

All numbers input are integers between -1000 and 1000.

Output Format

Print all the numbers before the cumulative sum become negative.

Sample Input

1

2

88

-100

49

Sample Output

1

2

88

Solution

```
// Online Java Compiler
// Use this editor to write, compile and run your Java code online
import java.util.*;
class HelloWorld {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
      //  System.out.println("Enter number");
      //  int s=sc.nextInt();
```

```
         int sum=0;
        while(true)
        {
            int a=sc.nextInt();
            sum=sum+a;
            if(sum<0)
            break;
            else
            System.out.println(a);
        }
}
}
```

26.Money from ATM

Pooja would like to withdraw X US dollar from an ATM. The cash machine will only accept the transaction if X is a multiple of 5, and Pooja's account balance has enough cash to perform the withdrawal transaction (including bank charges). For each successful withdrawal the bank charges 0.50 US dollar. Calculate Pooja's account balance after an attempted transaction.

Input Format

Positive integer X - the amount of cash which Pooja wishes to withdraw.

Nonnegative number Y with two digits of precision - Pooja's initial account balance.

Constraints

- 0 < X <= 2000
- 0<= Y <= 2000

Output Format

Output the account balance after the attempted transaction, given as a number with two digits of precision. If there is not enough money in the account to complete the transaction, output the current bank balance.

Sample Input 0

30 120.00

Sample Output 0

89.50

Explanation 0

Successful Transaction

Sample Input 1

42 120.00

Sample Output 1

120.00

Explanation 1

Incorrect Withdrawal Amount (not multiple of 5)

Sample Input 2

300 120.00

Sample Output 2

120.00

Explanation 2

Insufficient Funds

Sample Input 3

50 250.00

Sample Output 3

199.50

Explanation 3

Successful Transaction

Sample Input 4

500 400.00

Sample Output 4

400.00

Explanation 4

Insufficient Funds

Solution

```
import java.io.*;
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output
to STDOUT. Your class should be named Solution. */
        Scanner sc=new Scanner(System.in);
        double x=sc.nextDouble();
        double y=sc.nextDouble();
        if(x%5.00!=0 || x>y)
        {
            System.out.println(String.format("%.2f", y));
        }
        else
            System.out.println(String.format("%.2f",y-(x+0.50)));
    }
}
```

27.Total Expenses 1

While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If the quantity and price per item are input, write a program to calculate the total expenses.

Input Format

The first input is contains integers quantity and second input is contain price.

Constraints

- 1 ≤ quantity,price ≤ 100000

Output Format

Output the total expenses while purchasing items.

Sample Input 0

100 120

Sample Output 0

12000

Sample Input 1

10 20

Sample Output 1

200

Sample Input 2

1200 20

Sample Output 2

21600

Sample Input 3

120 400

Sample Output 3

48000

Sample Input 4

1500 30

Sample Output 4

40500

Solution

```
import java.io.*;
import java.util.*;
public class Solution {
public static void main(String[] args) {
      Scanner sc= new Scanner(System.in);
             int result = 0;
            int quantity = sc.nextInt();
```

```
            int price = sc.nextInt();
            result = quantity*price;
            if(quantity>1000){
                result = result -(result*10)/100;
                System.out.println(String.format("%d",result)) ;
            }
            else
                System.out.println(String.format("%d",result)) ;
    }
}
```

28. Sort 0 1 1

You are given an integer array A that contains only integers 0 and 1. Write a function to sort this array. Find a solution which scans the array only once. Don't use extra array. You need to change in the given array itself. So no need to return or print anything.

Input Format

- Line 1 : Integer N (Array Size)
- Line 2 : Array elements (separated by space)

Constraints

- 1 <= N <= 10^6

Output Format

Sorted array elements

Sample Input 0

7

0 1 1 0 1 0 1

Sample Output 0

0 0 0 1 1 1 1

Sample Input 1

6

0 1 0 1 0 1

Sample Output 1

0 0 0 1 1 1

Sample Input 2

8

0 1 1 1 0 1 0 1

Sample Output 2

0 0 0 1 1 1 1 1

Sample Input 3

10

0 1 1 0 1 0 1 1 0 0

Sample Output 3

0 0 0 0 0 1 1 1 1 1

Sample Input 4

9

0 1 1 0 1 0 1 0 0

Sample Output 4

0 0 0 0 0 1 1 1 1

SOLUTION

```
import java.io.*;
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output
to STDOUT. Your class should be named Solution. */
        Scanner sc=new Scanner(System.in);
```

```
        int n=sc.nextInt();
        int[] ar=new int[n];
        for(int i=0;i<n;i++)   {
            ar[i]=sc.nextInt();
        }
        Arrays.sort(ar);
          for(int i=0;i<n;i++) {
           System.out.print(ar[i]+" ");
        }}}
```

29. Minimum Length Word 1

Given a string S (that can contain multiple words), you need to find the word which has minimum length.

Note : If multiple words are of same length, then answer will be first minimum length word in the string.

Words are seperated by single space only.

Input Format

String S

Constraints

- 1 <= Length of String S <= 10^5

Output Format

Minimum length word

Sample Input 0

this is test string

Sample Output 0

is

Sample Input 1

cat eat rat

Sample Output 1

cat

Sample Input 2

health is wealth

Sample Output 2

is

Sample Input 3

ram and sham are brothers

Sample Output 3

ram

Sample Input 4

coding will help you for placement

Sample Output 4

you

30. T-primes 4

We know that prime numbers are positive integers that have exactly two distinct positive divisors. Similarly, we'll call a positive integer 't' T-prime, if t has exactly three distinct positive divisors.

You are given an array of n positive integers. For each of them determine whether it is T-prime or not.

Input Format

The first line contains a single positive integer, n, showing how many numbers are in the array. The next line contains n space-separated integers xi .

Constraints

- 1 ≤ n ≤ 10^5
- 1 ≤ xi ≤ 10^12

Output Format

Print n lines: the i-th line should contain "YES" (without the quotes), if number xi is T-prime, and "NO" (without the quotes), if it isn't.

Sample Input 0

3

4 5 6

Sample Output 0

YES

NO

NO

Explanation 0

The given test has three numbers. The first number 4 has exactly three divisors – 1, 2 and 4, thus the answer for this number is "YES". The second number 5 has two divisors (1 and 5), and the third number 6 has four divisors (1, 2, 3, 6), hence the answer for them is "NO".

Solution

```
import java.io.*;
import java.util.*;
public class Solution {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int n= scan.nextInt();
        int arr[]=new int[n];
        int count=0;
        for(int i=0;i<arr.length;i++)
        {
            arr[i]=scan.nextInt();
        }
        for(int i=0;i<arr.length;i++)
```

```
        {
            for(int j=1;j<=arr[i];j++)
            {
                if(arr[i]%j==0)
                {
                    count=count+1;
                }
            }
            if(count==3)
            {
                System.out.println("YES");
                count=0;
            }
            else
                System.out.println("NO");
            }
}
}
```

31.One of the important aspect of object oriented programming is readability of the code. To enhance the readability of code, developers write function and variable names in Camel Case. You are given a string, S, written in Camel Case.
FindAllTheWordsContainedInIt.

Input Format

A single line contains the string.

Constraints

|S|<=1000

Output Format

Print words present in the string, in the order in which it appears in the string.

Sample Input

IAmACompetitiveProgrammer

Sample Output

I

Am

A

Competitive

Programmer

Explanation

There are 5 words in the string.

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        String b=sc.next();
        //String[] b=s.split("");
        System.out.print(b.charAt(0));
        for(int i=1;i<b.length();i++)
        {
            char j=b.charAt(i);
            //String g="";
            if(Character. isLowerCase(j))
        System.out.print(j);
        else{
         System.out.println();
          System.out.print(j);
        }
        }
```

```
    }
}
```

32.Take as input S, a string. Write a function that does basic string compression. Print the value returned. E.g. for input “aaabbccds” print out a3b2c2d1s1.

Input Format

A single String S

Constraints

1 < = length of String < = 1000

Output Format

The compressed String.

Sample Input

aaabbccds

Sample Output

a3b2c2d1s1

Explanation

In the given sample test case 'a' is repeated 3 times consecutively, 'b' is repeated twice, 'c' is repeated twice and 'd and 's' occurred only once.

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
         int c=1,i;
        for( i=0;i<s.length()-1;i++)
        {
```

```
            if(s.charAt(i)==s.charAt(i+1))
            c++;
            else
            {System.out.print(s.charAt(i)+""+c);
            c=1;
            }
        }
        System.out.print(s.charAt(i)+""+c);

    }
}
```

33. A Good String is a string which contains only vowels (a,e,i,o,u) . Given a string S, print a single positive integer N where N is the length of the longest substring of S that is also a Good String.

Note: The time limit for this problem is 1 second, so you need to be clever in how you compute the substrings.

Input Format

A string 'S'

Constraints

Length of string < 10^5

Output Format

A single positive integer N, where N is the length of the longest sub-string of S that is also a Good String.

Sample Input

cbaeicde

Sample Output

3

Explanation

Longest good substring is "aei"

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        int max=0,c=0,i;
        s=s.toLowerCase();
        for( i=0;i<s.length();i++)
        {
            char a=s.charAt(i);
            if(a=='a' || a=='e' || a=='i' || a=='o' || a=='u')
            {
            c++;
            }
            else
            {
               if(c>max)
               max=c;
            c=0;
            }
        }
        if(i==s.length() && c>max)
         System.out.println(c);
         else
        System.out.println(max);
    }
}
```

34.Given a string, find the first non-repeating character in it. For example, if the input string is “coding blocks”, then the output should be ‘d’ and if the input string is “coding”, then the output should be ‘c’.

Input Format

The first line contains T denoting the number of testcases. Then follows description of testcases. Each case begins with a single integer N denoting the length of string. The next line contains the string S.

Constraints

String Length <100000

Output Format

For each testcase, print the first non repeating character present in string. Print -1 if there is no non repeating character.

Sample Input

4

codingblocks

abbac

java

ccdd

Sample Output

d

c

j

-1

Solution

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

```
        for(int i=0;i<k;i++)
        {
            int c=0;
            String s=sc.next();
            for(int j=0;j<s.length();j++)
            {
                c=0;
                for(int p=0;p<s.length();p++)
                {
                    if(s.charAt(j)==s.charAt(p))

                    c++;
                    // System.out.println("c"+" "+c);
                }
                if(c==1)
                {
                System.out.println(s.charAt(j));
                break;
                }

            }
             if(c>1)
  System.out.println("-1");


            }
        }

    }
```

35.Take as input S, a string. Write a function that replaces every even character with the character having just higher ASCII code and every odd character with the character having just lower ASCII code. Print the value returned.

Input Format

String

Constraints

Length of string should be between 1 to 1000.

Output Format

String

Sample Input

abcg

Sample Output

badf

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        String k="";
        for(int i=0;i<s.length();i++)
        {
            if(i%2==0)
            k=k+(char)(s.charAt(i)+1);
            else
            k=k+(char)(s.charAt(i)-1);
```

```
        }
        System.out.print(k);
    }
}
```

36. Take as input S, a string. Write a program that inserts between each pair of characters the difference between their ascii codes and print the ans.

Input Format

String

Constraints

Length of String should be between 2 to 1000.

Output Format

String

Sample Input

acb

Sample Output

a2c-1b

Explanation

For the sample case, the Ascii code of a=97 and c=99 ,the difference between c and a is 2.Similarly ,the Ascii code of b=98 and c=99 and their difference is -1. So the ans is a2c-1b.

Solution

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
```

```
        String k="";
        for(int i=0;i<s.length()-1;i++)
        {
            int d=s.charAt(i+1)-s.charAt(i);
            k=k+s.charAt(i)+d;
        }
        k=k+s.charAt(s.length()-1);
        System.out.println(k);
    }
}
```

37. Take as input S, a string. Write a function that returns the character with maximum frequency. Print the value returned.

Input Format

String

Constraints

A string of length between 1 to 1000.

Output Format

Character

Sample Input

aaabacb

Sample Output

a

Explanation

For the given input string, a appear 4 times. Hence, it is the most frequent character.

Solution

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

```
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        int a[]=new int[26];
        for(int i=0;i<s.length();i++)
        a[s.charAt(i)-97]++;
        int max=a[0],index=0;
        for(int j=0;j<26;j++)
        {
            if(a[j]>max)
            {
                max=a[j];
                index=j;
            }
        }

        System.out.print((char)(index+97));
    }
}
```

38.Array Merge

You are given two sorted arrays of integers, arr1 and arr2,

and two integers m and n indicating the length of arr1 and arr2 respectively.

Your task is to merge arr2 into arr1 in sorted order.

Write a Java program that takes two sorted arrays as input and returns the

merged sorted array.

```
public static int[] mergeArrays(int[] arr1,  int[] arr2){}
```

Input:

Two sorted arrays of integers, arr1 and arr2.

Output:

A sorted array containing elements from both arr1 and arr2.

Example:

Input:

arr1 = [1, 3, 5]

arr2 = [2, 4, 6]

Output:

[1, 2, 3, 4, 5, 6]

Constraints:

You must implement a function mergeArrays(int[] arr1, int[] arr2) to solve this problem.

The input arrays are sorted in non-decreasing order.

The length of arr1 is equal to m + n, where m is the length of the initialized part.

The length of arr2 is equal to n.

You can modify the input array arr1 in-place.

Test Cases:

Test Case 1:

Input:

arr1 = [1, 3, 5]

arr2 = [2, 4, 6]

Output:

[1, 2, 3, 4, 5, 6]

Test Case 2:

Input:

arr1 = [1, 2, 3]

arr2 = [4, 5, 6]

Output:

[1, 2, 3, 4, 5, 6]

Test Case 3:

Input:

arr1 = [1, 3, 5, 7, 9]

arr2 = [2, 4, 6]

Output:

[1, 2, 3, 4, 5, 6, 7, 9]

Test Case 4:

Input:

arr1 = [2, 4, 6]

m = 3

arr2 = [1, 3, 5]

n = 3

Output:

[1, 2, 3, 4, 5, 6]

Test Case 5:

Input:

arr1 = [1, 2, 3]

arr2 = [4, 5, 6]

Output:

[1, 2, 3, 4, 5, 6]

Test Case 6:

Input:

arr1 = [1, 2, 3]

arr2 = []

Output:

[1, 2, 3]

```
package pack3;

import java.util.Arrays;
```

```
import java.util.Scanner;

public class Tester {

	public static int[] mergeArrays(int[] arr1, int[] arr2) {

		int[] result = new int[arr1.length + arr2.length];
		int pos = 0;
		for (int i = 0; i < arr1.length; i++) {
			result[pos] = arr1[i];
			pos++;
		}

		for (int i = 0; i < arr2.length; i++) {
			result[pos] = arr2[i];
			pos++;
		}

		return result;
	}

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter length of array-1 ? ");
		int len = sc.nextInt();

		int[] arr1 = new int[len];
		System.out.println("Enter elements of array-1 ? ");
		for (int i = 0; i < arr1.length; i++) {
```

```
                arr1[i] = sc.nextInt();

            }

            System.out.println("Enter length of array-2 ? ");
            len = sc.nextInt();

            int[] arr2 = new int[len];
            System.out.println("Enter elements of array-2 ? ");
            for (int i = 0; i < arr2.length; i++) {
                arr2[i] = sc.nextInt();

            }
            int[] re = mergeArrays(arr1, arr2);
            Arrays.sort(re);
            System.out.println(Arrays.toString(re));
            sc.close();

        }
}
```

39.You are given an array of integers and an integer k. Your task is to rotate the array to the right by k steps.

Write a Java program that takes an array of integers and an integer k as input and returns the rotated array.

```
public static int[] rotateArray(int[] nums, int k){}
```

Input:

An array of integers, nums, of length n (1 <= n <= 10^6).

An integer k representing the number of steps to rotate the array (0 <= k <= 10^6).

Output:

An array of integers representing the rotated array.

Example:

Input:

nums = [1, 3, 5, 2, 2]

k = 2

Output:

[2, 2, 1, 3, 5]

Constraints:

You must implement a function rotateArray(int[] nums, int k) to solve this problem.

The input array may contain both positive and negative integers.

Test Cases:

Test Case 1:

Input:

nums = [1, 3, 5, 2, 2]

k = 2

Output:

[2, 2, 1, 3, 5]

Test Case 2:

Input:

nums = [1, 2, 3, 4, 5]

k = 3

Output:

[3, 4, 5, 1, 2]

Test Case 3:

Input:

nums = [10, -5, 8, -2, 3]

k = 1

Output:

[3, 10, -5, 8, -2]

Test Case 4:

Input:

nums = [-7, 1, 5, 2, -4, 3, 0]

k = 4

Output:

[2, -4, 3, 0, -7, 1, 5]

Test Case 5:

Input:

nums = [2, 2, 2, 2]

k = 2

Output:

[2, 2, 2, 2]

Test Case 6:

Input:

nums = [1]

k = 1

Output:

[1]

```
package pack3;
```

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

public class Tester {

	public static void rotateOnes(int[] input) {

		int temp = input[input.length - 1];
		for (int i = input.length - 2; i >= 0; i--) {

			input[i + 1] = input[i];

		}
		input[0] = temp;
	}

	public static int[] rotateArray(int[] nums, int k) {
		for (int i = 0; i < k; i++) {
			rotateOnes(nums);

		}
		return nums;
	}

	public static void main(String[] args) {

		Scanner sc = new Scanner(System.in);
		System.out.println("Enter length of array ? ");
		int len = sc.nextInt();
```

```
            int[] arr = new int[len];
            System.out.println("Enter elements of array ? ");
            for (int i = 0; i < arr.length; i++) {
                  arr[i] = sc.nextInt();

            }

            System.out.println("Enter no. of rotation ? ");
            int n = sc.nextInt();
            arr = rotateArray(arr, n);
            System.out.println(Arrays.toString(arr));

            sc.close();

      }
}
```

40.You are given a binary number in the form of a string containing only '0's and '1's. Your task is to convert this binary number to its equivalent decimal representation.

Write a Java program that takes a binary number as input and converts it to decimal.

Input:

A binary string, binaryNumber, consisting of '0's and '1's. (1 <= |binaryNumber| <= 10^6)

Output:

An integer representing the decimal equivalent of the given binary number.

Example:

Input:

binaryNumber = "1101"

Output: 13

Function Signature:

```
public static int binaryToDecimal(String binaryNumber){
                // your code here .....................
}
```

Constraints:

You must implement a function binaryToDecimal(String binaryNumber) to solve this problem.

Ensure your program handles large binary strings efficiently.

The input binary string will always be valid (contains only '0's and '1's).

Test Cases:

Test Case 1:

Input:

binaryNumber = "1010"

Output: 10

Test Case 2:

Input:

binaryNumber = "111"

Output: 7

Test Case 3:

Input:

binaryNumber = "1001001"

Output:

73

Test Case 4:

Input:

binaryNumber = "0"

Output:

0

Test Case 5:

Input:

binaryNumber = "1101111010100100100101"

Output:

3647781

Test Case 6:

Input:

binaryNumber = "11111111111111111111"

Output:

8388607

Test Case 7:

Input:

binaryNumber = "100000000000000000000000"

Output:

16777216

Make sure your program correctly converts binary strings to decimal numbers and meets the specified constraints.

**********Answers
****

```
package pack2;

import java.util.Scanner;

public class Taster {

	public static int binaryToDecimal(String binaryNumber)  {
		int len = binaryNumber.length() ;
		int sum = 0;
		for (int i = 0; i < len; i++) {
			int num = binaryNumber.charAt(len-1-i)-48;

			num = (int)(num*Math.pow(2, i));
			sum = sum + num;

		}
		return sum;

	}
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		String binaryNumber = sc.next();
		int op = binaryToDecimal(binaryNumber);
		System.out.println(op);
		sc.close();


	}

}
```

41.Count the Number of Possible Triangles

You are given an array of positive integers.

Your task is to find and count the number of possible triangles that can be formed using elements

from the array.

Write a Java program that takes an array of positive integers as input and returns the count of possible triangles.

public static int countPossibleTriangles(int[] input) {}

Input:

An array of positive integers, nums, of length n (3 <= n <= 1000).

Each element in the array is a positive integer (1 <= nums[i] <= 1000).

Output:

An integer representing the count of possible triangles.

Example:

Input: arr= {4, 6, 3, 7}

Output: 3

Explanation: There are three triangles

possible {3, 4, 6}, {4, 6, 7} and {3, 6, 7}.

Note that {3, 4, 7} is not a possible triangle.

Input: arr= {10, 21, 22, 100, 101, 200, 300}

Output: 6

Explanation: There can be 6 possible triangles:

{10, 21, 22}, {21, 100, 101}, {22, 100, 101},

{10, 100, 101}, {100, 101, 200} and {101, 200, 300}

Constraints:

You must implement a function countPossibleTriangles(int[] nums) to solve this problem.

The input array will contain at least three elements.

The elements of the array are positive integers.

Test Cases:

Test Case 1:

Input:

nums = [3, 4, 5, 6]

Output:

4

Test Case 2:

Input:

nums = [1, 2, 3]

Output:

0

Test Case 3:

Input:

nums = [5, 7, 8, 10, 12]

Output:

9

Test Case 4:

Input:

nums = [10, 21, 22, 100]

Output:

1

Test Case 5:

Input:

nums = [4, 6, 3, 7]

Output:

3

Test Case 6:

Input:

nums = [8, 4, 6, 3, 7]

Output:

8

```
package pack3;

public class Tester {
	public static int countPossibleTriangles(int[] input) {

		int count = 0;
		for (int i = 0; i < input.length; i++) {
			for (int j = i+1; j < input.length; j++) {
				for (int k = j+1; k < input.length; k++) {
					int a = input[i];
					int b = input[j];
					int c = input[k];
					if( ((a+b) > c)  ) count++;
					//if( ((a+b) > c) && ((b+c) > a) && ((c+a)
> b) ) count++;

				}
			}
		}

		return count;
	}

	public static void main(String[] args) {

		System.out.println(countPossibleTriangles(new int[] {1, 2,
3}));
	}
```

```
}
```

42.Distinct Elements in String Array

You are given an array of strings. Your task is to find the distinct elements in the array.

Write a Java program that takes an array of strings as input and returns a set containing the distinct elements.

```
public static String[] findDistinctElements(String[] input) {}
```

Input:

An array of strings, strArray, of length n (1 <= n <= 10^6).

Output:

A set containing the distinct elements in the array.

Example:

Input:

```
strArray = ["apple", "orange", "banana", "apple", "grape", "banana", "apple", "orange", "grape", "kiwi", "kiwi", "kiwi"]
```

Output:

```
{"apple", "orange", "banana", "grape", "kiwi"}
```

In the example, the distinct elements are "apple", "orange", "banana", "grape", and "kiwi".

Constraints:

You must implement a function findDistinctElements(String[] strArray) to solve this problem.

The input array may contain strings with both uppercase and lowercase letters.

Test Cases:

Test Case 1:

Input:

strArray = ["apple", "orange", "banana", "apple", "grape", "banana", "apple", "orange", "grape", "kiwi", "kiwi", "kiwi"]

Output:

{"apple", "orange", "banana", "grape", "kiwi"}

Test Case 2:

Input:

strArray = ["apple", "orange", "banana", "kiwi", "orange", "kiwi", "apple"]

Output:

{"apple", "orange", "banana", "kiwi"}

Test Case 3:

Input:

strArray = ["hello", "world", "hello", "java", "world", "java", "java", "python"]

Output:

{"hello", "world", "java", "python"}

Test Case 4:

Input:

strArray = ["apple", "APPLE", "Orange", "orange", "Banana", "BANANA", "banana"]

Output:

{"apple", "APPLE", "Orange", "orange", "Banana", "BANANA", "banana"}

Test Case 5:

Input:

strArray = ["abc", "ABC", "abc", "ABC", "AbC", "aBC"]

Output:

{"abc", "ABC", "AbC", "aBC"}

```
package pack3;

import java.util.Arrays;
import java.util.Scanner;

public class Tester {

	public static String[] findDistinctElements(String[] input) {

		String[] result = new String[input.length];
		int pos = 0;
		for (int i = 0; i < input.length; i++) {
			boolean f = true;
			String s1 = input[i];
			for (int j = 0; j < result.length; j++) {
				String s2 = result[j];
```

```
                        if (s1.equals(s2)) {
                              f = false;
                              break;
                        }

                  }
                  if (f) {
                        result[pos] = s1;
                        pos++;
                  }

            }

            String[] s = new String[pos];
            for (int i = 0; i < pos; i++) {
                  s[i] = result[i];

            }

            return s;
      }

      public static void main(String[] args) {

//          String[] strArray = { "apple", "orange", "banana", "apple",
"grape", "banana", "apple", "orange", "grape",
//                        "kiwi", "kiwi", "kiwi" };
//          System.out.println(Arrays.toString(strArray));
//          strArray = findDistinctElements(strArray);
//          System.out.println(Arrays.toString(strArray));
```

```
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter length of  Array");
            int len  = sc.nextInt();
            String[] strArray = new String[len];
            for (int i = 0; i < len; i++) {
                  strArray[i] = sc.next();

            }
            strArray = findDistinctElements(strArray);
            System.out.println(Arrays.toString(strArray));
            sc.close();


      }
}
```

44.Find First and Last Position of Element in Sorted Array

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

```
class Solution {
    public static int[] searchRange(int[] nums, int target) {

    }
}
```

Example 1:

Input: nums = [5,7,7,8,8,10], target = 8

Output: [3,4]

Example 2:

Input: nums = [5,7,7,8,8,10], target = 6

Output: [-1,-1]

Example 3:

Input: nums = [], target = 0

Output: [-1,-1]

Constraints:

0 <= nums.length <= 10^5

-10^9 <= nums[i] <= 10^9

nums is a non-decreasing array.

-10^9 <= target <= 10^9

Test case 1

Input: nums = [1,1,1,1,1], target = 1

Output: [0,4]

Test case 2

Input: nums = [0,1,7,8,9,9], target = 9

Output: [4,5]

```
package pack3;
import java.util.*;
public class Tester {

      public static int countVal(int[] input, int val) {
            int count = 0;
```

```
        for (int i = 0; i < input.length; i++) {
            if (input[i] == val)
                count++;

        }
        return count;
    }

     public static int[] searchRange(int[] nums, int target) {
          int count = countVal(nums, target);

          int[] res = new int[2];
          res[0] = -1;
          res[1] = -1;
          int pos = 1;
          for (int i = 0; i < nums.length; i++) {
               if( nums[i]== target && pos == 1) {
                    res[0] = i;
                    pos++;
               }

               else if( nums[i]== target && pos == count) {
                    res[1] = i;
                    break;
               }

               else if(nums[i]== target) {pos++;}
```

```
            }

             return res;

           }

      public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            System.out.print("Enter Length Of array ?");
            int len = sc.nextInt();
            int[] input = new int[len];
            System.out.print("Enter Elements Of array ?");
            for (int i = 0; i < input.length; i++) {
                  input[i] = sc.nextInt();

            }
            System.out.print("Enter Target Value ?");
            int target = sc.nextInt();

            int[] result = searchRange(input, target);
            Arrays.sort(result);
            System.out.println(Arrays.toString(result));
            sc.close();

      }

}
```

45.You are given a pair of coordinates (x, y), where 'x' and 'y' are integers. Your task is to determine in which quadrant of the Cartesian coordinate system the point lies. The Cartesian coordinate system is divided into four quadrants: Quadrant I, Quadrant II, Quadrant III, and Quadrant IV.

Write a Java program that takes the values of 'x' and 'y' as input and determines the quadrant in which the point lies.

Input:

Two integers, x and y, representing the coordinates of a point. (-10^6 <= x, y <= 10^6)

Output:

A string indicating the quadrant in which the point lies:

Quadrant I if the point is in the first quadrant (x > 0, y > 0)

Quadrant II if the point is in the second quadrant (x < 0, y > 0)

Quadrant III if the point is in the third quadrant (x < 0, y < 0)

Quadrant IV if the point is in the fourth quadrant (x > 0, y < 0)

Origin if the point is at the origin (x = 0, y = 0)

Axes if the point is on one of the coordinate axes (x = 0 or y = 0)

```
public static String findQuadrant(int x, int y){
 /////// your code here
}
```

Example:

Input:

x = 4

y = 3

Output: Quadrant I

Input:

x = -2

y = 5

Output: Quadrant II

Input:

x = -3

y = -2

Output:

Quadrant III

Constraints:

You must implement a function findQuadrant(int x, int y) to solve this problem.

Ensure that the program correctly determines the quadrant and handles points at the origin or on the axes.

The values of 'x' and 'y' will be integers within the specified range.

Test Cases:

Test Case 1:

Input:

x = 5

y = 0

Output: Axes

Test Case 2:

Input:

x = -7

y = -9

Output: Quadrant III

Test Case 3:

Input:

x = 0

y = 0

Output:

Origin

Test Case 4:

Input:

x = 2

y = -4

Output:

Quadrant IV

Test Case 5:

Input:

x = -6

y = 8

Output:

Quadrant II

Test Case 6:

Input:

x = 0

y = 7

Output:

Axes

```
package pack2;

import java.util.Scanner;
```

```
public class Hello {

	public static String findQuadrant(int x, int y) {
		if (x>0 && y>0) {
			return "Quadrant I";
		} else if(x < 0 && y > 0) {
			return "Quadrant II";
		} else if(x < 0 && y < 0) {
			return "Quadrant III";
		} else if(x > 0 && y < 0) {
			return "Quadrant IV";
		} else if(x == 0 && y == 0) {
			return "Origin";
		}else if(x == 0 || y == 0) {
			return "Axes";
		}
		return "";
	}
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		int x = sc.nextInt();
		int y = sc.nextInt();
		String op = findQuadrant(x,y);
		System.out.println(op);
		sc.close();



	}

}
```

46.LCM

Take the following as input.

A number (N1)

A number (N2)

Write a function which returns the LCM of N1 and N2. Print the value returned.

Input Format

Constraints

0 < N1 < 1000000000

0 < N2 < 1000000000

Output Format

Sample Input

4

6

Sample Output

12

Explanation

The smallest number that is divisible by both N1 and N2 is called the LCM of N1 and N2.

Test cases:-

Input	Output
12 44	132
4 9	36
44 92	1104

3 9	9
23 35	805

```
import java.util.Scanner;



public class LCM {
    public static void main(String[] args) {



        Scanner sc=new Scanner(System.in);
        int num1=sc.nextInt();
        int num2=sc.nextInt();
        lcm(num1, num2);




    }
    public static void lcm(int num1,int num2){
         if(num1>num2){
        int temp = num2;
        num2=num1;
        num1=temp;
    }



    int lcm=num2;
    while(lcm>0){
        if(lcm%num2==0 && lcm%num1==0){
```

```
            break;
        }
        lcm++;
    }
    System.out.println(lcm);



    }
}
```

48.You are given two positive integers, A and B. Your task is to find their Highest Common Factor (HCF),

also known as the Greatest Common Divisor (GCD).

The HCF of two numbers is the largest positive integer that divides both A and B without

leaving a remainder.

Write a Java program that takes two integers as input and calculates their HCF.

```
public static long calculateHCF(long a, long b){
  // your code here only
}
```

Input:

Two positive integers A and B where (1 <= A, B <= 10^9).

Output:

An integer representing the HCF of A and B.

Example:

Input:

A = 12

B = 18

Output:

6

Constraints:

You must implement a function calculateHCF(int A, int B) to solve this problem.

Your program should be able to handle large inputs efficiently.

Do not use any external libraries or built-in HCF/GCD functions.

Test Cases:

Test Case 1:

Input:

A = 15

B = 20

Output: 5

Test Case 2:

Input:

A = 8

B = 12

Output:

4

Test Case 3:

Input:

A = 21

B = 14

Output:

7

Test Case 4:

Input:

A = 100

B = 200

Output:

100

Test Case 5:

Input:

A = 35

B = 56

Output:

7

Test Case 6:

Input:

A = 1

B = 1

Output:

1

Test Case 7:

Input:

A = 999999999

B = 999999998

Output:

1

```
package pack2;
import java.util.Scanner;
```

```
public class Tester {
	public static long calculateHCF(long a, long b) {
	    long hcf =  a<b ? a : b ;
	    while (true) {

	     if(a % hcf == 0 && b % hcf == 0) break;
	     hcf--;

	     }

	    return hcf;
	}

	public static void main(String[] args) {
	     Scanner sc = new Scanner(System.in);
	     long a = sc.nextLong();
	     long b = sc.nextLong();
	     long op = calculateHCF(a, b);
	     System.out.println(op);
	     sc.close();
	}

}
```

49.Nth Fibonacci

Take N as input. Print Nth Fibonacci Number, given that the first two numbers in the Fibonacci Series are 0 and 1.

Input Format

Constraints

0 <= N <= 1000

Output Format

Sample Input

10

Sample Output

55

Explanation

The 0th fibonacci is 0 and 1st fibonacci is 1.

Test cases:-

Input	Output
14	377
8	21
20	6765
19	4181
13	233

```
import java.util.*;
public class Main {
public static void main(String args[]) {
// Your Code Here
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        fib(n);


}
```

```
    public static void fib(int n){
        int a=0;
        int b=1;
        int c=0;
        for(int i=2;i<=n;i++){
            c=a+b;
            a=b;
            b=c;
        }
        System.out.println(c);
    }
}
```

50.Check Prime

Take as input a number N, print "Prime" if it is prime if not Print "Not Prime".

Input Format

Constraints

2 < N <= 1000000000

Output Format

Sample Input

3

Sample Output

Prime

Explanation

The output is case specific

Test Cases:-

Input	Output
6	Not Prime
1212	Not Prime

23	Prime
57	Prime
12345	Not Prime

Solution:-

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



    }
    public static void prime(int n){
        boolean isprime=true;



        for(int i=2;i<=n/2;i++){
            if(n%i==0){
                isprime=false;
                break;
            }
        }
        if(isprime){
            System.out.println("Prime");
        }else{
            System.out.println("Not Prime");
        }
    }
```

}

51.Is Armstrong Number

Take the following as input.

A number

Write a function which returns true if the number is an Armstrong number and false otherwise, where the Armstrong number is defined as follows.

A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

abcd… = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ….

1634 is an Armstrong number as 1634 = 1^4 + 6^4 + 3^4 + 4^4

371 is an Armstrong number as 371 = 3^3 + 7^3 + 1^3

Input Format

Single line input containing an integer

Constraints

0 < N < 1000000000

Output Format

Print boolean output for each test case.

"true" if the given number is an Armstrong Number, else print "false".

Sample Input

371

Sample Output

true

Explanation

Use functions. Write a function to check if the number is an Armstrong number or not. Numbers are Armstrong if it is equal to the sum of each digit raised to the power of the number of digits.

Test cases:-

Input	Output
153	True

245	False
567	False
407	True
123	False

Solution:-

```
import java.util.*;
public class Main {
    public static void main(String args[]) {



        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        Armstrong(n);
    }
    public static void Armstrong(int n){



        int temp=n;
        int NoOfDig=0;
        while(temp>0){
            NoOfDig++;
            temp/=10;
        }
        temp=n;
        int sum=0;
        while(temp>0){
            int digit=temp%10;
            sum+=Math.pow(digit,NoOfDig);
```

```
            temp=temp/10;



        }
        if(sum==n){
            System.out.println("true");
        }else{
            System.out.println("false");
        }
    }
}
```