*/\* Q:- Armstrong Number (Easy)*

*------------------------------------------------------------------*

*Give an integer (x), return {true} if (x) is an armstrong number*

*and {false} otherwise.*

*-------------------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

import java.util.*\**;

public class armstrongNum {

    static boolean getarmstrongNum(int x){

        int dup=x;

        int sum=0;

        while (x>0) {

            int lastDigit=x%10;

            sum+=(lastDigit\*lastDigit\*lastDigit);

            x=x/10;

        }

        if (dup==sum) return true;

        return false;

    }

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.println("Enter a number to check weather it is armstrong number or not:");

        int x=in.nextInt();

        System.out.println(getarmstrongNum(x));

    }

}

*/\* Q:- Reverse an Integer (Easy)*

*----------------------------------------------------------------------------------------*

*Given a number (n) Count the number if digits in (n) which evenly divide (n).Return*

*an integer , total number of digit of (n) which divides (n) evenly.*

*-----------------------+++++++++++-------------------------------------------------------*

*Note: Evenly divides means wheather (n) is divisible by a digit ,i.e leaves a remainder*

*0 when divided. Time Complexity :O(n) and Space Complexity:-O(1);*

*-----------------------------------------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

import java.util.*\**;

public class digit\_ext {

    static int getdigitExtraction(int n){ *//method 1*

        int Count=0;

        while (n>0) {

*// int Last\_digit=n%10;      // it might be extra and act as useless var*

*// return Last\_digit;               // Space Complexity:- O(1)*

            Count ++;                           *//Time Complexity:- O(n)*

            n=n/10;

        }

        return Count;

    }

    static int getExt(int n){  *//method 2*

        int count=(int)(Math.log10(n)+1);

        return count;

    }

    public static void main(String[] args) {

        Scanner in=new Scanner(System.in);

        System.out.println("Enter a number:");

               int n=in.nextInt();

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

    }

}

*/\* Q:- Print All Divisions. also calcuate the sum of the factors. (GFG)*

*----------------------------------------------------------*

*Given an integer (x) , return the all factors of n.*

*----------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

import java.util.*\**;

public class Division {

    static void getDivisors(int x){              *// method 1*

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

            if (x%i==0) {

                System.out.println(i);

            }

        }

    }

    static void getdividor(int x){              *// method 2*

        for (int i = 1; i <= Math.sqrt(x); i++) {

             if (x%i==0) {

                System.out.println(i);

                if (x/i!=i) {

                    System.out.println(x/i);

                }

             }

        }

    }

    static void getfactorSum(int x){

        int sum=0;

        for (int i = 1; i <= Math.sqrt(x); i++) {

            if (x%i==0) {

                sum+=i;

                if (x/i!=i) {

                    sum+=x/i;

                }

            }

        }

        System.out.println(sum);

    }

    public static void main(String[] args) {

        Scanner in =new Scanner(System.in);

        System.out.println("Enter a number:");

        int x=in.nextInt();

       getfactorSum(x);

    }

}

*/\* Q:- LCM and GCD && \*\*\*\*Euclidean ALGO\*\*\*\**

*-------------------------------------------------------------------*

*Given two integer a and b return the LCM and GCD .*

*-------------------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

import java.util.*\**;

public class lcmandGCD {

    static void getGCD(int a,int b){

        int gcd=1;                      *//Method 1 {Time Complexity:- O(min(a,b))}*

        for (int i = 1; i < a; i++) {

            if (a%i==0 && b%i==0) {

                gcd=i;

            }

        }

        System.out.println(gcd);

    }

    static void getGCDM1(int a,int b){     *//Method 2 {Time Complexity:- O(min(a,b))}*

        for (int i = Math.min(a, b); i >=1 ; i-- ) {

            if (a%i==0 && b%i==0) {

                System.out.println(i);

                break;

            }

        }

    }

    static void euclideanApproch(int a,int b){          *// Euclidean Approch*

        while (a>0 && b>0) {                 *// Time Complexity:-O(log (Fi) min(a,b))*

            if (a>b) {

                a=a%b;

            }

            else{

                b=b%a;

            }

        }

        if (a==0) {

            System.out.println(b);

        }

        else{

            System.out.println(a);

        }

    }

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.println("Enter two number:");

        int a=in.nextInt();

        int b=in.nextInt();

        euclideanApproch(a, b);    }

}

*/\*Q:- Prime Number*

*-------------------------------------------------------------------------------------------*

*Given an integer (n) and returns true if the number is prime and false otherwise.*

*-------------------------------------------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

import java.util.*\**;

public class Prime {

    static boolean getPrime(int n){

        int count=0;                        *//method 1 { Time Complexity:- O(n)}*

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

            if (n%i==0)  count++;

        }

        if (count==2) return true;

        return false;

    }

    static boolean getprime(int n){

        int count=0;

        for (int i = 1; i <= Math.sqrt(n); i++) {

            if (n%i==0)                      *//Method 2 { Time Complexity:- O(Sqrt(n))}*

            {

                count++;

            if ((n/i) != i)

            {

                count++;

            }

        }

    }

        if (count==2) return true;

        return false;

    }

    public static void main(String[] args) {

        Scanner in =new Scanner(System.in);

        System.out.println("Enter a number:");

        int n=in.nextInt();

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

    }

}

*/\* Q:- Reverse Integer (Medium)*

*-------------------------------------------------------------------------------------------------*

*Given a signed 32-bit integer(x),return(x)with its digits*

*reversed. If reversing (x) causes the value to go outside the*

*signed 32-bit integer range [-2^31,2^31-1],then return 0.*

*--------------------------------------------------------------------------------------------------*

*Note:- Assume the environment does not allow you to store 64-bit integers(Signed or Unsigned).*

*--------------------------------------------------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

 import java.util.*\**;

public class reverseDigit {

    static int getReverse(int x){

        int rev=0;

        while (x>0) {

            int lastDigit=x%10;

            x=x/10;

            rev=(rev\*10)+lastDigit;

        }

        return rev;

    }

    public static void main(String[] args) {

        Scanner in =new Scanner(System.in);

        System.out.println("Enter a number:");

        int x=in.nextInt();

        System.out.println(getReverse(x));

    }

}

*/\* Q:- Palindrome Number*

*--------------------------------------------------------------------------------*

*Given am integer (n) return true if (n) is palindrome no and false otherwise.*

*--------------------------------------------------------------------------------*

*\*/*

package Maths\_for\_DSA;

import java.util.*\**;

public class palindome {

    static boolean getPalindrome(int n){

        int dup=n;

        int rev=0;

        while (n>0) {

           int lastDigit=n%10;

           rev=(rev\*10)+lastDigit;

           n=n/10;

        }

        if (rev==dup) return true;

        return false;

    }

    public static void main(String[] args) {

        Scanner in =new Scanner(System.in);

        System.out.println("Enter a number:");

        int n=in.nextInt();

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

    }

}