1295. Find Numbers with Even Number of Digits 
Easy 1270 Addt0List Share 
Given an array nums of integers, return how many of them contain an even number of digits. 
Example 1: 
- (12, 
Input: nums - 
Output : 2 
Explanation: 
12 contains 2 digits (even number of digits). 
345 contains 3 digits (Odd number Of digits). 
2 contains 1 digit (odd number of digits) . 
contains 1 digit (odd number of digits) . 
7896 contains 4 digits (even number Of digits) . 
Therefore only 12 and 7896 contain an even number Of digits. 

Ans:

class Solution {

public int findNumbers(int[] nums) {

int c=0;

for(int num : nums)

{

if(countDigit(num)%2==0)

{

c++;

}

}

return c;

}

public int countDigit(int num)

{

int c=0;

while(num>0)

{

c++;

num /= 10;

}

return c;

}

}

1672. Richest Customer Wealth 
Easy 1846 Q 261 Add to List [fi Share 
You are given an m x n integer grid accounts where is the amount of money the ith customer has in the 5th bank. Return 
the wealth that the richest customer has. 
A customers wealth is the amount of money they have in all their bank accounts. The richest customer is the customer that has the maximum 
wealth. 
Example 1: 
Input: accounts 
Output : S 
Explanation: 
1st customer has wealth - 
2nd customer has wealth 
Both customers are considered 
the 
richest with a wealth of 6 each, 
so return 6. 

class Solution

{

public int maximumWealth(int[][] accounts)

{

int max =0;

int sum =0;

for(int r=0; r < accounts.length; r++)

{

for(int c=0; c < accounts[r].length; c++)

{

sum = sum + accounts[r][c];

}

if(sum > max)

{

max = sum;

}

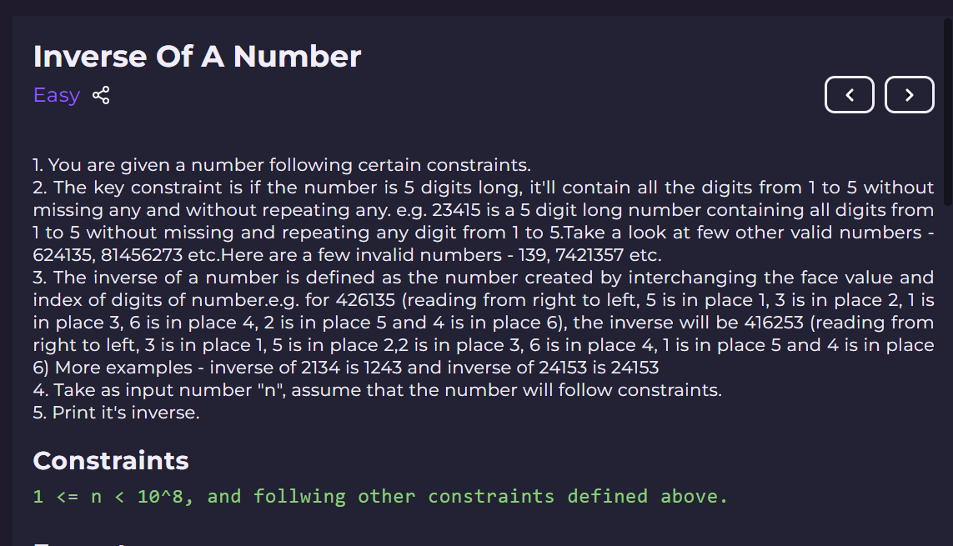
sum=0;

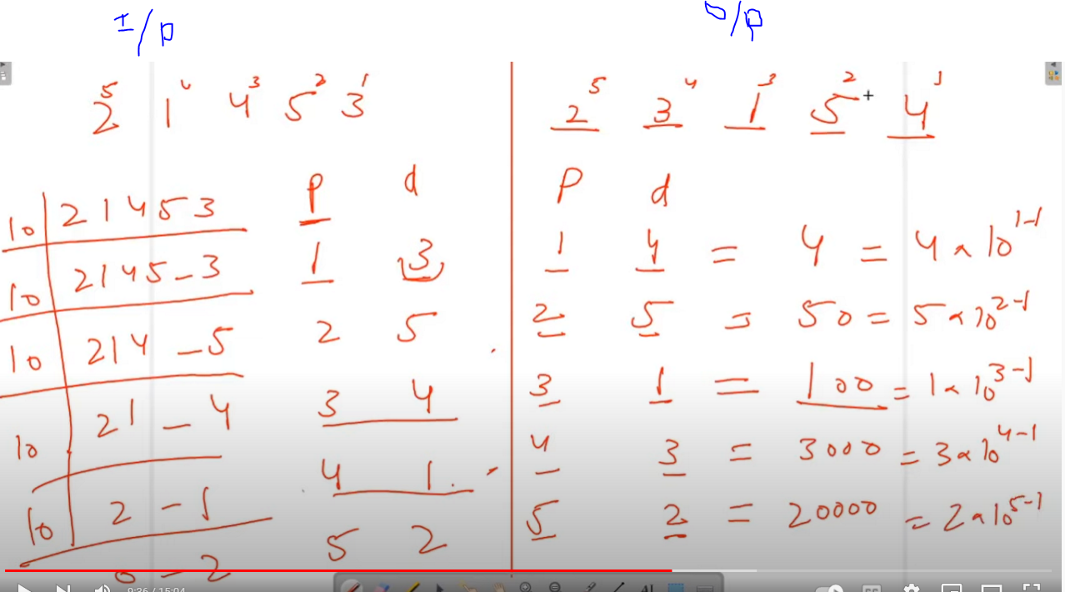
} // End outer loop

return max;

}

}// class end





import java.util.\*;

public class Main

{

public static void main(String[] args)

{

  Scanner in = new Scanner(System.in);

   int number = in.nextInt();

  int invertNumb = 0;

  int digitExtracted =0;

  //Position of digit ,as we divide the number ,will increase till n =0

  int postnDigit =1;

   //extarct ecah digit ,till last

   while(number>0)

   {

       // get each digit from back

       digitExtracted = number%10;

       //Reduce one for postion in power

       digitExtracted--;

       invertNumb = invertNumb + (int) (postnDigit \* Math.pow(10 , digitExtracted));

        // increase the position of Digit

        postnDigit++;

      //Reduce The number

      number = number/10;

   }

   System.out.print(invertNumb);

 }

}

A screenshot of a computer

Description automatically generated

Explanation :

Suppose n= 20, bulbs were initially off.

toggling is also haapening n times i.e. 20 times bulbs will toggle.

First time : all 20 bulbs will be toogled

Second time : every second bulb will be toggled.

..... and so on,

we have to find after nth toggle ,which bulbs be ON.

A screenshot of a whiteboard with writing

Description automatically generated

Every Perfect Square, have odd number, of Factors.

Solution: Print perfect square of n , since only Perfect square number will be on after nth fluctuations.

import java.util.Scanner;

public class Main{

    public static void main(String[] args)

    {

        // Write your code here

        Scanner sc = new Scanner(System.in);

        int number = sc.nextInt();

        for(int i = 1 ; i\*i <= number; i++)

        {

// print perfect square numbers

            System.out.println(i\*i);

        }

    }

}

A screenshot of a number prime

Description automatically generated

import java.util.\*;

public class Prime

{

public static void main(String[] args)

{

Scanner scn = new Scanner(System.in);

int t = scn.nextInt();

int i;

int c = 0;

while(t > 0)

{

int n = scn.nextInt();

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

{

if(n % i == 0)

{

c = 1;

break;

}

}

if(c !=0)

{

System.out.println("not prime");

}

else

{

System.out.println("prime");

}

c=0;

t--;

}

}

}

A screenshot of a computer

Description automatically generated

public class PrimeTillN {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int low = sc.nextInt();

int high = sc.nextInt();

int count =0;

// 2 <= low < high < 10 ^ 6

while(low <= high)

{

// run loop from 2 to till square root of every number coming in low

for(int i=2; i\*i <= low; i++)

{

//check for divisibility

if(low % i == 0)

{

//increment a count variable and break

count ++;

break;

}

}

if(count ==0)

{

System.out.println(low);

}

// increase low

low++;

//reset the count

count =0;

}

}

}

A screenshot of a computer

Description automatically generated

import java.util.Scanner;

public class Fibo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int first =0;

int last =1;

//all three variables act as pointer , move one step ahead in each step

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

{

//print Fib no

System.out.println(first);

// third pointer is found by adding prev two term

int current= first + last;

//assign last ,to first i.e. move 1 step ahead

first = last;

// assign current to last i.e. move 1 step ahead

last = current;

}

}

}

Time Complexity : O(n), since loop will run n times

A screenshot of a computer

Description automatically generated

import java.util.Scanner;

public class LCMGCD {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n1 = sc.nextInt();

int n2 = sc.nextInt();

int original\_n1 = n1;

int original\_n2 = n2;

int temp;

while(n2 % n1 !=0)

{

// get the reminder, make it new divisor

temp = n2 % n1;

//assign new dividend

n2 = n1;

//assign remainder for divisor

n1 =temp;

}

System.out.println(n1);

// n1 \* n2 = lcm \* gcd

System.out.println( original\_n1\*original\_n2/n1);

}

}

Time Complexity:

O(min(a, b)) Since we have just one loop from min(a, b) to 1.

A screenshot of a computer

Description automatically generated

import java.util.Scanner;

public class PrimeFactor {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n1 = sc.nextInt(); // .i.e 10

//run loop till square root of number

for(int i=2; i\*i <= n1; i++)

{

// keep dividing num, till further it's not divisible

while(n1 % i == 0) // 10/1 ..2..

{

n1 = n1 / i;

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

}

}

// Factor may lie beyond square root of n ,e.g 26

if(n1 !=1)

{

System.out.println(n1);

}

}

}

Time Complexity : O(sqrt(n));

**Digit Frequency**

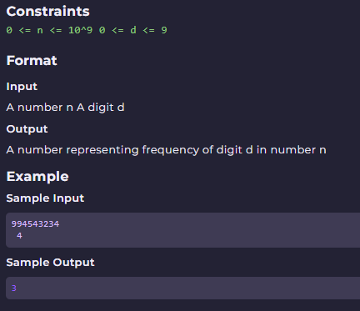
Easy

1. Each digit (0 to 9) denotes the student of the Optica Student community.

2. You are given a number n where ith digit denotes that ith task that is assigned to the corresponding digit student.

2. You are given a digit d denotes a student.

3. You are required to calculate the frequency of digit d in number n or how many tasks are assigned to student d.



import java.util.\*;

public class Main {

    public static void main(String[] args) {

        Scanner scn = new Scanner(System.in);

        int n = scn.nextInt();

        int d = scn.nextInt();

        int f = getDigitFrequency(n, d);

        System.out.println(f);

    }

    public static int getDigitFrequency(int n, int d)

{

        int counter = 0;

        int rem =0;

        while(n>0)

        {

            // extract the digit

            rem = n%10;

            //compare with target digit, if found incement the counter

            if(rem==d)

            {

                counter++;

            }

            //reduce the digit

            n =n/10;

        }

        return counter;

    }

}

A screenshot of a computer program

Description automatically generated

A whiteboard with numbers and arrows

Description automatically generated

A white board with blue writing on it

Description automatically generated

import java.util.\*;

  public class Main

{

  public static void main(String[] args)

{

      Scanner scn = new Scanner(System.in);

      int n = scn.nextInt();// number

      int b = scn.nextInt();// base

      int dn = getValueInBase(n, b);

      System.out.println(dn);

   }

   public static int getValueInBase(int n, int b)

{

       int co =0;

       int c =1;

       int rem=0;

       while(n>0)

       {

           //get the digit one by one

           rem = n % b;

           // place the digit to correct place

           co = rem \* c + co;

           //reduce the digit

           n = n / b;

           //increase the c

           c = c \* 10;

       }

       return co;

   }

 }

TIME COMPLEXITY :

We are extracting digits of number n and performing some minute calculations, which will take O(log10 n) time as there can be maximum floor(log10 n) digits in a number n.

A screenshot of a computer

Description automatically generated

Soln:

We'll be converting 1172 (base 8) into a decimal number.

A screenshot of a computer

Description automatically generated

import java.util.\*;

  public class Main

{

  public static void main(String[] args) {

      Scanner scn = new Scanner(System.in);

      int n = scn.nextInt();

      int b = scn.nextInt();

      int d = getValueIndecimal(n, b);

      System.out.println(d);

   }

   public static int getValueIndecimal(int n, int b)

{

      int rem =0;

      int decimalValue =0;

      int powerIncrementor =1;

      while(n>0)

      {

         // extract the digit one by one

         rem = n%10;

         // form the decimal value

         decimalValue = rem\*powerIncrementor + decimalValue;

         // incement the power

         powerIncrementor = powerIncrementor\*b;

         //reduce the digit

         n = n/10;

      }

      return decimalValue;

   }

 }