**ASSIGNMENTS:**

**Count of divisors**

import java.util.\*;

class GFG{

static void countIndex(int []A, int N)

{

int MAX = Arrays.stream(A).max().getAsInt();

int []freq = new int[MAX + 1];

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

freq[A[i]]++;

int []res = new int[MAX + 1];

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

for (int j = i; j <= MAX; j += i) {

if (i == j) {

res[i] += (freq[j] - 1);

}

else {

res[i] += freq[j];

res[j] += freq[i];

}

}

}

for (int i = 0; i < N; i++) {

System.out.print(res[A[i]]+ " ");

}

}

public static void main(String[] args)

{

int []A = { 2, 3, 4, 5, 6 };

int N = A.length;

countIndex(A, N);

}

}

**Prime Sum**

#include <stdio.h>

#include <stdbool.h>

bool SieveOfEratosthenes(int n, bool isPrime[])

{

isPrime[0] = isPrime[1] = false;

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

isPrime[i] = true;

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

if (isPrime[p] == true) {

for (int i = p \* p; i <= n; i += p)

isPrime[i] = false;

}

}

}

void findPrimePair(int n)

{ bool isPrime[n + 1];

SieveOfEratosthenes(n, isPrime);

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

if (isPrime[i] && isPrime[n - i]) {

printf("%d %d",i,n-i);

return;

}

}

}

// Driven program

int main()

{

int n;

scanf("%d",&n);

findPrimePair(n);

return 0;

}

**Lucky Numbers**

#include <stdio.h>

#define bool int

bool isLucky(int n)

{

static int counter = 2;

if (counter > n)

return 1;

if (n % counter == 0)

return 0;

int next\_position = n - (n / counter);

counter++;

return isLucky(next\_position);

}

// Driver code

int main()

{

int x;

scanf("%d",&x);

if (isLucky(x))

printf("%d is a lucky no.", x);

else

printf("%d is not a lucky no.", x);

getchar();

}import java.util.Arrays;

import java.util.Vector;

class GFG

{

static int primeCount(int arr[], int n)

{

int max\_val = Arrays.stream(arr).max().getAsInt();

Boolean[] prime = new Boolean[max\_val + 1];

for (int i = 0; i < max\_val + 1; i++)

{

prime[i] = true;

}

prime[0] = false;

prime[1] = false;

for (int p = 2; p \* p <= max\_val; p++)

{

if (prime[p] == true)

{

for (int i = p \* 2; i <= max\_val; i += p)

{

prime[i] = false;

}

}

}

int count = 0;

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

{

if (prime[arr[i]])

{

count++;

}

}

return count;

}

// Driver code

public static void main(String[] args)

{

int arr[] = {1, 2, 3, 4, 5, 6, 7};

int n = arr.length;

System.out.println(primeCount(arr, n));

}

}

Compute nCr%m

import java.util.\*;

class Main{

static int computeBinomialCoefficientsModuloP(int n, int r, int p)

{

int C[] = new int[r+1];

C[0] = 1;

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

{

for (int j = Math.min(i, r); j >0 ; j--)

{

C[j] = (C[j - 1] + C[j])%p;

}

}

return C[r];

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int k = sc.nextInt();

int p = sc.nextInt();

int val = computeBinomialCoefficientsModuloP(n, k, p);

if(val != 0)

System.out.println(val);

else

System.out.println(0);

}

}

Compute nCr %p

import java.util.\*;

public class Main

{

static int fact(int N){

if(N==1 || N==0) return 1;

else return N\*fact(N-1);

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int r=sc.nextInt();

int c=sc.nextInt();

System.out.println(((fact(n)/((fact(n-r)\*fact(r)))%c)));

}

}

Delete one

import java.io.\*;

class Deletion {

static int deleteElement(int arr[], int n, int x)

{

int i;

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

if (arr[i] == x)

break;

if (i < n)

{

n = n - 1;

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

arr[j] = arr[j+1];

}

return n;

}

public static void main(String[] args)

{

int arr[] = {11, 15, 6, 8, 9, 10};

int n = arr.length;

int x = 6;

n = deleteElement(arr, n, x);

System.out.println("Modified array is");

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

System.out.print(arr[i]+" ");

}

}

Enumerating GCD

#include <stdio.h>

int gcd(int m,int n){

if(m==n) return m;

else return 1;

}

int main()

{

int M,N;

scanf("%d",&M,&N);

printf("GCD : %d",gcd(M,N));

return 0;

}

Greatest Common Divisor

#include <stdio.h>

int gcd(int A, int B) {

return B==0 ? A : gcd(B, A%B);}

int main()

{

int M,N;

scanf("%d",&M,&N);

printf("GCD : %d",gcd(M,N));

return 0;

}

Pubg

def gcd(self, A, B):

while B:

A, B = B, A % B

return A

def solve(self, A):

A\_length = len(A)

if A\_length == 1: return A[0]

min\_health = self.gcd(A[0], A[1])

for ele in A:

min\_health = self.gcd(min\_health, ele)

return min\_health