**DIA**

1)**Naughty Kid in a Staircase**

**// You are using Java**

**import java.util.Scanner;**

**class Staircase{**

**static int kid(int n){**

**if(n<4){**

**return n;**

**}**

**else{**

**return kid(n-2)+kid(n-1);**

**}**

**}**

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

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

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

**System.out.print(kid(n));**

**}**

**}**

**2)Robot Maneuvering**

**// You are using Java**

**import java.util.Scanner;**

**class Robot{**

**public static int robo(int r,int c){**

**if(r==1||c==1){**

**return 1;**

**}**

**else{**

**return robo(r-1,c)+robo(r,c-1);**

**}**

**}**

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

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

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

**int c=ab.nextInt();**

**System.out.print(robo(r,c));**

**}**

**}**

**3)Coin change problem**

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

**class GFG {**

**static int count(int coins[], int n, int sum)**

**{**

**if (sum == 0)**

**return 1;**

**if (sum < 0)**

**return 0;**

**if (n <= 0)**

**return 0;**

**return count(coins, n - 1, sum)**

**+ count(coins, n, sum - coins[n - 1]);**

**}**

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

**{**

**int coins[] = { 1, 2, 3 };**

**int n = coins.length;**

**System.out.println(count(coins, n, 4));**

**}**

**}**

**4)palindrome string**

import java.util.\*;

class Main

{

public static void main(String args[])

{

String str="";

Scanner s=new Scanner(System.in);

str=s.nextLine();

boolean flag= palindromeCheck(str);

System.out.println(flag);

}

public static boolean palindromeCheck(String s)

{

if(s.length() == 0 || s.length() == 1)

{

return true;

}

if(s.charAt(0) == s.charAt(s.length()-1))

{

return palindromeCheck(s.substring(1, s.length()-1));

}

return false;

}

}

5.Sieve of Eratosthenes

import java.util.\*;

class main{

void display (int n){

boolean prime[]=new boolean[n+1];

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

prime[i]=true;

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

if(prime[j]==true){

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

prime[i]=false;

}

}

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

if(prime[i]==true){

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

}

}

}

public static void main(String[] args){

Scanner my=new Scanner(System.in);

int a=my.nextInt();

main s=new main();

s.display(a);

}

}

6. Sieve of Sundaram

import java.util.\*;

class main{

static int sieve(int n){

int New=(n-1)/2;

boolean marked[]=new boolean[New+1];

Arrays.fill(marked,false);

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

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

marked[i+j+2\*i\*j]=true;

if(n>2)

System.out.print(2+" ");

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

if(marked[i]==false)

System.out.print(2\*i+1+" ");

return -1;

}

public static void main(String[]ags){

Scanner my=new Scanner (System.in);

int a=my.nextInt();

sieve(a);

}

}

7. Toggle the Bulbs

import java.util.\*;

class main{

public static void main(String[]args){

Scanner my=new Scanner (System.in);

int a =my.nextInt();

double b=Math.sqrt(a);

double round=Math.round(b\*100)/100;

System.out.printf("%.0f",round);

}

}

8. Euclidean Algorithm

import java.util.\*;

class main{

public static void main(String[]arfs){

Scanner my=new Scanner(System.in);

int x=my.nextInt();

int y=my.nextInt();

int gcd=1;

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

if(x%i==0&&y%i==0){

gcd=i;

}

}

System.out.print("G.C.D of "+x+" and "+y+" is "+gcd);

}

}

9. Emirp number

import java.util.Scanner ;

class Main

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in) ;

int n = sc.nextInt() ;

int temp = 0 ;

while(n>0)

{

temp = temp\*10 + n%10 ;

n /= 10 ;

}

n = temp ;

for(int i = 2 ; i <= (int)Math.sqrt(n) ; i++)

{

if(n%i == 0)

{

System.out.print("Not Emirp") ;

System.exit(0) ;

}

}

System.out.print("Emirp") ;

}

}

10. Euler's Totient

import java.util.Scanner ;

class Main

{

static int phi(int n)

{

int result = n;

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

{

if (n % p == 0)

{

while (n % p == 0)

n /= p;

result -= result / p;

}

}

if (n > 1)

result -= result / n;

return result;

}

public static void main (String[] args)

{

Scanner sc = new Scanner(System.in) ;

int n = sc.nextInt() ;

System.out.print( phi(n) ) ;

}

}

11. Prime numbers Sum

import java.io.\*;

import java.util.\*;

class Main

{

static ArrayList<Integer> set =

new ArrayList<Integer>();

static ArrayList<Integer> prime =

new ArrayList<Integer>();

static boolean isPrime(int x)

{

int sqroot = (int)Math.sqrt(x);

if (x == 1)

return false;

for (int i = 2;

i <= sqroot; i++)

if (x % i == 0)

return false;

return true;

}

static void display()

{

int length = set.size();

for (int i = 0;

i < length; i++)

System.out.print(

set.get(i) + " ");

System.out.println();

}

static void primeSum(int total, int N,

int S, int index)

{

if (total == S &&

set.size() == N)

{

display();

return;

}

if (total > S ||

index == prime.size() || set.size() >= N)

return;

set.add(prime.get(index));

primeSum(total + prime.get(index),

N, S, index + 1);

set.remove(set.size() - 1);

primeSum(total, N,

S, index + 1);

}

static void allPrime(int N,

int S, int P)

{

for (int i = P + 1;

i <= S ; i++)

{

if (isPrime(i))

prime.add(i);

}

if (prime.size() < N)

return;

primeSum(0, N, S, 0);

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in) ;

int N = sc.nextInt();

int P = sc.nextInt();

int S = sc.nextInt();

allPrime(N, S, P);

}

}

12. Segmented Sieve of Eratosthenes

import java.util.\*;

class Main

{

static void simpleSieve(int limit, Vector<Integer> prime)

{

boolean mark[] = new boolean[limit+1];

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

mark[i] = true;

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

{

if (mark[p] == true)

{

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

mark[i] = false;

}

}

for (int p=2; p<limit; p++)

{

if (mark[p] == true)

{

prime.add(p);

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

}

}

}

static void segmentedSieve(int n)

{

int limit = (int) (Math.floor(Math.sqrt(n))+1);

Vector<Integer> prime = new Vector<>();

simpleSieve(limit, prime);

int low = limit;

int high = 2\*limit;

while (low < n)

{

if (high >= n)

high = n;

boolean mark[] = new boolean[limit+1];

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

mark[i] = true;

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

{

int loLim = (int) (Math.floor(low/prime.get(i)) \* prime.get(i));

if (loLim < low)

loLim += prime.get(i);

for (int j=loLim; j<high; j+=prime.get(i))

mark[j-low] = false;

}

for (int i = low; i<high; i++)

if (mark[i - low] == true)

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

low = low + limit;

high = high + limit;

}

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in) ;

int n = sc.nextInt() ;

segmentedSieve(n);

}

}

13-->Linear Search

import java.util.Scanner;

class LinearSearch{

static void linearSearch(int[] arr,int size,int key){

boolean found=false;

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

if(arr[i]==key){

found=true;

break;

}

}

if(found){

System.out.print("Yes");

}else{

System.out.print("No");

}

}

public static void main(String[] args){

Scanner scan=new Scanner(System.in);

int num=scan.nextInt();

int arr[]=new int[num];

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

arr[i]=scan.nextInt();

}

int key=scan.nextInt();

linearSearch(arr,num,key);

}

}

14-->Bubble Sort

import java.util.Scanner;

class BubbleSort{

static void bubbleSort(int[] arr){

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

for(int j=0;j<arr.length-1;j++){

if(arr[j]>arr[j+1]){

int temp=arr[j];

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

arr[j+1]=temp;

}

}

}

}

public static void main(String[] args){

Scanner scan=new Scanner(System.in);

int size=scan.nextInt();

int arr[]=new int[size];

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

arr[i]=scan.nextInt();

}

bubbleSort(arr);

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

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

}

}

}

15--> Binary Search

import java.util.Scanner;

class BinarySearch{

static void binarySearch(int[] arr,int size,int key){

int first=0;

int last=size-1;

int mid=(first+last)/2;

while(first<=last){

if(arr[mid]>key){

last=mid-1;

}else if(arr[mid]<key){

first=mid+1;

}else{

System.out.print(mid);

break;

}

mid=(first+last)/2;

}

if(first>last){

System.out.print("NO OCCURRENCES");

}

}

public static void main(String[] args){

Scanner scan=new Scanner(System.in);

int size=scan.nextInt();

int arr[]=new int[size];

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

arr[i]=scan.nextInt();

}

int key=scan.nextInt();

binarySearch(arr,size,key);

}

}

16-->Merge Sort

import java.util.Scanner;

class MargeSort{

public static void mergeSort(int[] a,int n){

if(n<2){

return;

}

int mid=n/2;

int[] l=new int[mid];

int[] r=new int[n-mid];

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

l[i]=a[i];

}

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

r[i-mid]=a[i];

}

mergeSort(l,mid);

mergeSort(r,n-mid);

merge(a,l,r,mid,n-mid);

}

static void merge(int[] a,int[] l,int[] r,int left,int right){

int i=0,j=0,k=0;

while(i<left && j<right){

if(l[i]<=r[j]){

a[k++]=l[i++];

}else{

a[k++]=r[j++];

}

}

while(i<left){

a[k++]=l[i++];

}

while(j<right){

a[k++]=r[j++];

}

}

public static void main(String[] args){

Scanner scan=new Scanner(System.in);

int size=scan.nextInt();

int[] arr=new int[size];

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

arr[i]=scan.nextInt();

}

mergeSort(arr,size);

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

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

}

}

}

17-->Selection Sort

import java.util.Scanner;

class SelectionSort{

static void display(int[] arr){

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

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

}

System.out.println();

}

static void selectionSort(int[] arr){

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

// start=i;

for(int j=i+1;j<arr.length;j++){

if(arr[i]>arr[j]){

int temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

if(i!=arr.length-1){

System.out.print("Height order of students after iteration "+(i+1)+" \n");

display(arr);

}else{

System.out.println("After final comparison of all students , the height order becomes:");

display(arr);

}

}

}

public static void main(String[] args){

Scanner scan=new Scanner(System.in);

int size=scan.nextInt();

int arr[]=new int[size];

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

arr[i]=scan.nextInt();

}

System.out.print("Student's height order before sorting:\n");

display(arr);

selectionSort(arr);

}

}

18. String Sorting

import java.util.\*;

class Main

{

//static int MAX = 100;

public static void sortStrings(String[] arr, int n)

{

String temp;

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

{

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

{

if (arr[j].compareTo(arr[i]) > 0)

{

temp = arr[j];

arr[j] = arr[i];

arr[i] = temp;

}

}

}

}

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

String[] arr =new String[n];

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

{

arr[i]=sc.next();

}

sortStrings(arr, n);

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

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

}

}

import java.io.\*;

import java.util.\*;

class GFG {

static int SIZE = 10;

static void sortMat(int mat[][], int n)

{

int temp[] = new int[n \* n];

int k = 0;

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

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

temp[k++] = mat[i][j];

Arrays.sort(temp);

k = 0;

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

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

mat[i][j] = temp[k++];

}

static void printMat(int mat[][], int n)

{

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

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

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

System.out.println();

}

}

public static void main(String args[])

{

Scanner my=new Scanner(System.in);

int n=my.nextInt();

int mat[][] = new int[n][n];

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

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

mat[i][j]=my.nextInt();

}

}

sortMat(mat, n);

printMat(mat, n);

}

}

20-->Sorting Student's Data

import java.util.\*;

class data{

int roll;

String name;

int m1,m2,m3;

float avg;

int sum;

data(int num,String name,int m1,int m2,int m3,float avg,int sum){

this.roll=num;

this.name=name;

this.m1=m1;

this.m2=m2;

this.m3=m3;

this.avg=avg;

this.sum=sum;

}

}

class main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

data[] d=new data[n+1];

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

int num=sc.nextInt();

String name=sc.next();

int m1=sc.nextInt();

int m2=sc.nextInt();

int m3=sc.nextInt();

float avg=(float)(m1+m2+m3)/3;

int sum=m1+m2+m3;

d[i]=new data(num,name,m1,m2,m3,avg,sum);

}

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

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

if(d[i].avg<d[j].avg){

data t=d[i];

d[i]=d[j];

d[j]=t;

}

}

}

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

System.out.println(d[i].roll+" "+d[i].name+" "+d[i].sum+" "+d[i].avg);

}

}

}

21-->Insertion Sort

import java.util.\*;

public class InsertionSort {

static void sort(int arr[])

{

int n = arr.length;

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

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

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

j = j - 1;

}

arr[j + 1] = key;

}

}

static void printArray(int arr[])

{

int n = arr.length;

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

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

System.out.println();

}

public static void main(String args[])

{

Scanner scan=new Scanner(System.in);

int n=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

sort(arr);

printArray(arr);

}

};

22->Heap Sort

import java.util.Scanner;

public class HeapSort {

public static void sort(int arr[])

{

int N = arr.length;

for (int i = N / 2 - 1; i >= 0; i--)

heapify(arr, N, i);

for (int i = N - 1; i > 0; i--) {

int temp = arr[0];

arr[0] = arr[i];

arr[i] = temp;

heapify(arr, i, 0);

}

}

public static void heapify(int arr[], int N, int i)

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < N && arr[l] > arr[largest])

largest = l;

if (r < N && arr[r] > arr[largest])

largest = r;

if (largest != i) {

int swap = arr[i];

arr[i] = arr[largest];

arr[largest] = swap;

heapify(arr, N, largest);

}

}

public static void printArray(int arr[])

{

int N = arr.length;

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

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

System.out.println();

}

public static void main(String args[])

{

Scanner scan=new Scanner(System.in);

int n=scan.nextInt();

int arr[]=new int[n];

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

arr[i]=scan.nextInt();

}

sort(arr);

printArray(arr);

}

* }

23)**Naive Algorithm**

import java.util.Scanner;

class Main{

public static void main(String args[]){

Scanner scan = new Scanner(System.in);

String word = scan.nextLine();

String subWord = scan.nextLine();

if(word.indexOf(subWord) >= 0){

System.out.print("Found at " + word.indexOf(subWord) );

return;

}

System.out.print("Not Found");

}

}

24)**KMP Algorithm**

**import java.util.Scanner;**

**class Kmp{**

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

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

**String word = scan.nextLine();**

**String subWord = scan.nextLine();**

**boolean flag = false;**

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

**if(word.charAt(i) == subWord.charAt(0)){**

**int j=0;**

**for(j=0 ; j<subWord.length() ; j++){**

**if(word.charAt(j+i) != subWord.charAt(j)){**

**break;**

**}**

**}**

**if(j == subWord.length()){**

**System.out.println("Found at " + i);**

**flag = true;**

**}**

**}**

**}**

**if(!flag){**

**System.out.print("Not Found");**

**}**

**}**

**}**

**25)Rail fence Algorithm**

**import java.util.Scanner ;**

**class Main**

**{**

**static String Encryption(String plainText, int depth)**

**{**

**if(depth<2)**

**return plainText ;**

**int r = depth ;**

**int c = plainText.length() ;**

**char mat[][] = new char[r][c] ;**

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

**{**

**for(int j = 0 ; j < c ; j++)**

**{**

**mat[i][j] = '\*' ;**

**}**

**}**

**boolean dir\_down = false ;**

**int row = 0 , col = 0 ;**

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

**{**

**if (row == 0 || row == r-1)**

**dir\_down = !dir\_down ;**

**mat[row][col++] = plainText.charAt(i) ;**

**if(dir\_down)**

**row++ ;**

**else**

**row-- ;**

**}**

**char result[] = new char[c] ;**

**int k = 0 ;**

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

**{**

**for (int j=0; j < c ; j++)**

**{**

**if (mat[i][j] != '\*')**

**result[k++] = mat[i][j] ;**

**}**

**}**

**String encryptedString = new String (result) ;**

**return encryptedString ;**

**}**

**static String Decryption(String cipherText, int depth)**

**{**

**if(depth<2)**

**return cipherText ;**

**int r = depth ;**

**int c = cipherText.length() ;**

**char mat[][] = new char[r][c] ;**

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

**{**

**for(int j = 0 ; j < c ; j++)**

**{**

**mat[i][j] = '\*' ;**

**}**

**}**

**boolean dir\_down = false ;**

**int row = 0 , col = 0 ;**

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

**{**

**if (row == 0 || row == r-1)**

**dir\_down = !dir\_down ;**

**mat[row][col++] = '$' ;**

**if(dir\_down)**

**row++ ;**

**else**

**row-- ;**

**}**

**int k = 0 ;**

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

**{**

**for (int j = 0 ; j < c ; j++)**

**{**

**if (mat[i][j] == '$')**

**mat[i][j] = cipherText.charAt(k++) ;**

**}**

**}**

**char result[] = new char[c] ;**

**k = 0 ;**

**dir\_down = false ;**

**row = col = 0 ;**

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

**{**

**if (row == 0 || row == r-1)**

**dir\_down = !dir\_down ;**

**result[k++] = mat[row][col++] ;**

**if(dir\_down)**

**row++ ;**

**else**

**row-- ;**

**}**

**String decryptedString = new String(result) ;**

**return decryptedString ;**

**}**

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

**{**

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

**int depth ;**

**String plainText,cipherText,decryptedText ;**

**plainText = sc.next() ;**

**depth = sc.nextInt() ;**

**cipherText = Encryption(plainText, depth) ;**

**System.out.println("Encrypted text is: " + cipherText) ;**

**decryptedText = Decryption(cipherText, depth);**

**System.out.println("Decrypted text is: " + decryptedText) ;**

**}**

**}**

**26)Coin Calculator**

**// You are using Java**

**import java.util.Scanner;**

**class Calculator{**

**static void coincalc(int n){**

**}**

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

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

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

**int [] coin=new int[] {5,3,2,1};**

**int [] result=new int[4];**

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

**result[i]=n/coin[i];**

**n-=result[i]\*coin[i];**

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

**}**

**}**

**}**

**27)Rat in a Maze:**

**// You are using Java**

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

**class Maze{**

**static boolean isSafe(int [][]maze,int i,int j){**

**boolean flag=false;**

**if(i>=0 && i<maze.length&&j>=0&&j<maze[0].length && maze[i][j]==1){**

**flag=true;**

**}**

**return flag;**

**}**

**static boolean solveMaze(int [][]maze,int i,int j,int [][]sol){**

**if(i==maze.length-1 && j==maze[0].length-1 && maze[i][j]==1){**

**sol[i][j]=1;**

**return true;**

**}**

**if(isSafe(maze,i,j)){**

**sol[i][j]=1;**

**if(solveMaze(maze,i+1,j,sol)){**

**return true;**

**}**

**if(solveMaze(maze,i,j+1,sol)){**

**return true;**

**}**

**sol[i][j]=0;**

**return false;**

**}**

**return false;**

**}**

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

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

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

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

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

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

**maze[i][j]=ab.nextInt();**

**}**

**}**

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

**if(solveMaze(maze,0,0,sol)){**

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

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

**System.out.print(sol[i][j]+" ");**

**}**

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

**}**

**}**

**else{**

**System.out.print("Solution doesn't exist");**

**}**

**}**

**}**

**28)Longest Common Subsequence(LCS)**

**// You are using Java**

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

**class LongestCommonSubsequence {**

**int lcs( char[] X, char[] Y, int m, int n )**

**{**

**if (m == 0 || n == 0)**

**return 0;**

**if (X[m-1] == Y[n-1])**

**return 1 + lcs(X, Y, m-1, n-1);**

**else**

**return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));**

**}**

**int max(int a, int b) {**

**if (a > b)**

**{**

**return a;**

**}**

**return b;**

**}**

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

**{**

**LongestCommonSubsequence lcs = new LongestCommonSubsequence();**

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

**String s1 = sc.nextLine();**

**String s2 = sc.nextLine();**

**char[] X=s1.toCharArray();**

**char[] Y=s2.toCharArray();**

**int m = X.length;**

**int n = Y.length;**

**System.out.println(lcs.lcs( X, Y, m,n));**

**}**

**}**

**29)Implementation of Segment Tree**

**import java.util.Scanner ;**

**public class Main {**

**static int N = 100000;**

**static int n;**

**static int []tree = new int[2 \* N];**

**static void build( int []arr)**

**{**

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

**tree[n + i] = arr[i];**

**for (int i = n - 1; i > 0; --i)**

**tree[i] = tree[i << 1] +**

**tree[i << 1 | 1];**

**}**

**static void updateTreeNode(int p, int value)**

**{**

**tree[p + n] = value;**

**p = p + n;**

**for (int i = p; i > 1; i >>= 1)**

**tree[i >> 1] = tree[i] + tree[i^1];**

**}**

**static int query(int l, int r)**

**{**

**int res = 0;**

**for (l += n, r += n; l < r;**

**l >>= 1, r >>= 1)**

**{**

**if ((l & 1) > 0)**

**res += tree[l++];**

**if ((r & 1) > 0)**

**res += tree[--r];**

**}**

**return res;**

**}**

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

**{**

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

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

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

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

**a[i] = sc.nextInt() ;**

**build(a);**

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

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

**System.out.println(query(start, end));**

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

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

**updateTreeNode(index, element);**

**System.out.println(query(start, end));**

**}**

**}**

**30. Floyd Warshall Algorithm**

**// You are using Java**

**import java.util.Scanner;**

**class FloydWarshall {**

**final static int INF = 999 ;**

**static int V ;**

**static void floydWarshall(int graph[][])**

**{**

**int dist[][] = new int[V][V];**

**int i, j, k;**

**for (i = 0; i < V; i++)**

**for (j = 0; j < V; j++)**

**dist[i][j] = graph[i][j];**

**for (k = 0; k < V; k++) {**

**for (i = 0; i < V; i++) {**

**for (j = 0; j < V; j++) {**

**if (dist[i][k] + dist[k][j]< dist[i][j])dist[i][j]= dist[i][k] + dist[k][j];**

**}**

**}**

**}**

**printSolution(dist);**

**}**

**static void printSolution(int dist[][])**

**{**

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

**for (int j = 0; j < V; ++j) {**

**if (dist[i][j] == INF)**

**System.out.print("INF ") ;**

**else**

**System.out.print(dist[i][j] + " ") ;**

**}**

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

**}**

**}**

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

**{**

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

**V = sc.nextInt() ;**

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

**int graph[][] = new int [V][V] ;**

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

**for(int j = 0 ; j < V ; j++){**

**if(i==j)**

**graph[i][j] = 0 ;**

**else**

**graph[i][j] = INF ;**

**}**

**}**

**int start, end, value ;**

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

**start = sc.nextInt() ;**

**end = sc.nextInt() ;**

**value = sc.nextInt() ;**

**graph[start][end] = value ;**

**graph[end][start] = value ;**

**}**

**System.out.println("Original matrix") ;**

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

**for(int j = 0 ; j < V ; j++){**

**if(graph[i][j] == INF)**

**System.out.print("INF ") ;**

**else**

**System.out.print(graph[i][j]+" ") ;**

**}**

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

**}**

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

**System.out.println("Shortest path matrix") ;**

**floydWarshall(graph);**

**}**

**}**