Practice 1

QUESTION 1:
An array/list A which consists of n positive natural numbers, Your task is to find the inversion count in the given array/list A
Brief editorial on inversion count:
    In an array/list, inversion count indicates how far or close the array/list is from being sorted.
    If array/list is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum.
Note: Two elements A[i] and A[j] form an inversion if A[i] > A[j] and i < j.
Sample test case 1 is:
Input :
5 ----------> n
20 40 10 30 50 --------------> Elements in Array/List A
Output:
3 --------------> Output as required
Explanation to the sample test case 2: The array/list A: 20, 40, 10, 30, 50 has 3 inversions (20, 10), (40, 10), (40, 30).

Ans:

```
package q14175;
import java.io.*;
import java.util.*;

public class CTJ14175 {
public static void main(String args[]){
        Scanner s=new Scanner(System.in);
        int num=s.nextInt();
        int count=0;
        int a[]=new int[num];
        for(int i=0; i<num; i++){
                int n=s.nextInt();
                a[i]=n;
        }

        for(int i=0; i<num; i++){
                for(int j=i+1; j<num; j++){
                        if(a[i]>a[j]){count++;}
                }
        }

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

QUESTION 2:
Franky is working on a stack problem where two integer arrays push[]

and pop[] both of size n are given to him.
Both arrays consist of distinct values. As a result Franky has to print 1 i.e. true if this could have been the result of a sequence of push and pop operations on an initially empty stack, or print 0 i.e. false otherwise.
Sample test case 1:
Input:
5 ----------> n
1 2 3 4 5 ---------> Values in push[] array.
4 5 3 2 1 ---------> Values in pop[] array
pushed = [1,2,3,4,5], popped = [4,5,3,2,1]
Output:
1 ----------> i.e. true
Editorial: The following sequence can be created:
push(1),push(2), push(3), push(4),pop() value 4, push(5), pop() value 5, pop() value 3, pop() value 2, pop() value 1

Ans:

```
package q14344;
import java.io.*;
import java.util.*;
public class CTJ14344 {
public static void main(String args[]){
        Scanner s=new Scanner(System.in);
        int num=s.nextInt();
        int pushed[]=new int[num];
        int poped[]=new int[num];

        for(int i=0; i<num; i++){
                int a=s.nextInt();
                pushed[i]=a;
        }

        for(int j=0; j<num; j++){
                int b=s.nextInt();
                poped[j]=b;
        }
        Stack<Integer> s1=new Stack<>();

        for(int k=0, p=0; k<pushed.length; k++){
                s1.push(pushed[k]);


        while(s1.size() !=0 && s1.peek()==poped[p]){
                s1.pop();
                p++;
        }
        }
        if(s1.size()==0){System.out.println("1");}
        else{System.out.println("0");}

}
}
```

QUESTION 3:
An array A of size n is a Mount Everest if the following properties hold:
  n >= 3
      There exists some i with 0 < i < n - 1 such that:
    A[0] < A[1] < ... < A[i - 1] < A[i] A[i] > A[i + 1] > ... > A[n - 1]
Given a Mount Everest array A, return the index i such that A[0] < A[1] < ... < A[i - 1] < A[i] > A[i + 1] > ... > A[n - 1].
You must solve it in O(log(n)) time complexity.
Sample test case 1:
Input:
3 ----------> n i.e. A.length
0 1 0 --------------> A[]
Output:
1

ANS:

```
#include <iostream>

using namespace std;

int main()
{
        int n, index = -1;
        cin >> n;
        int a[n];
        for(int i = 0; i < n; i++)
                cin >> a[i];

        int l = 0, r = n - 1;
        while(l <= r)
        {
                int i = l + (r - l) / 2;
                if(i > 0 && i < n - 1 && a[i] > a[i - 1] && a[i] > a[i + 1])
                {
                        index = i;
                        break;
                }
                else if(i > 0 && a[i] > a[i - 1] || a[i] < a[i + 1])
                        l = i + 1;
                else
                        r = i - 1;
        }

        cout << index << endl;
        return 0;
}
```

QUESTION 4:

Given an array of integers buildingHeights of size n, where each value is representing the building height's standing adjacent to each other with share outer wall's, where the width of each building is 1, return the area of the largest rectangle in the building's formation.

Lets see one example:

You can see buildingHeights are: 2, 1, 5, 6, 2, 3. But the largest rectangle formation is highlighted in red, We can see buildings with height 5, and 6 are part of the largets rectange with rectangle dimention's height = 5 and width = 2, therefore area will be 5 x 2 = 10. Their can be many other rectangle formations exists, but the one with largest area highlighted above is 10.

Sample test case:
6 ------------> Number of buildings.
2 1 5 6 2 3 ------------> Building heights
10 ------------> Area of largests rectange formed.

ANS:

```
#include <iostream>
#include <algorithm>
#include <stack>

using namespace std;

int main()
{
        int n;
        cin >> n;
        int h[n];
        for(int i = 0; i < n; i++)
                cin >> h[i];

        stack <int> st;
        int lsmall[n], rsmall[n];
        for(int i = 0; i < n; i++)
        {
                while(!st.empty() && h[st.top()] >= h[i])
                        st.pop();
                if(st.empty())
                        lsmall[i] = 0;
                else
                        lsmall[i] = st.top() + 1;
                st.push(i);
        }

        while(!st.empty())
                st.pop();

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

```
                while(!st.empty() && h[st.top()] >= h[i])
                        st.pop();
                if(st.empty())
                        rsmall[i] = n - 1;
                else
                        rsmall[i] = st.top() - 1;
                st.push(i);
        }

        int maxarea = 0;
        for(int i = 0; i < n; i++)
                maxarea = max(maxarea, h[i] * (rsmall[i] - lsmall[i]
+ 1));

        cout << maxarea << endl;
        return 0;
}
```

QUESTION 5:
Jack is working on an array of integers A of size n and an integer k.
As a result he has to print the number of unique k-diff pairs in the array.
A k-diff pair is an integer pair (A[i], A[j]), where the following are true:
    0 <= i, j < n
    i != j
    A[i] - A[j] == k
Notice that |val| denotes the absolute value of val.
Sample test case 1:
Input:
5 ------------------ n
3 1 4 1 5 ---------------- A[]
2 ----------------- k
Output:
2

ANS:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
        int n;
        int k;
        cin>>n;
        int arr[n];
        for(int i=0;i<n;i++){
        cin>>arr[i];
        }
        cin>>k;
        set<pair<int,int>> res;
        map<int,int> mp;
        mp[arr[0]]++;
        for(int i=1;i<n;i++){
```

```
                if(mp.find(arr[i]+k) !=mp.end()){
                        res.insert(make_pair(arr[i],arr[i]+k));
                }
                if(mp.find(arr[i]-k) != mp.end()){
                        res.insert(make_pair(arr[i],arr[i]-k));
                }
                mp[arr[i]]++;
        }

        cout<<res.size();
        return 0;
}
```

QUESTION 6:
Franky has given an integer array A of size n and an integer k.
For each index i where 0 <= i < A.length, change A[i] to be either A[i] + k or A[i] - k.
The score of A is the difference between the maximum and minimum elements in A.
Franky has to print the minimum score of A after changing the values at each index.
Sample test case 1:
Input:
1 ----- n
1 ------ A[]
0 ------------ k
Output:
 0

 ANS:
 #include <bits/stdc++.h>

```
using namespace std;

int main()
{
        int n, k, score;
        cin >> n;
        int a[n];
        for(int i = 0; i < n; i++)
                cin >> a[i];
        cin >> k;

        if(n == 1)
                score = 0;
        else
        {
                sort(a, a + n);
                int low = 0, high = 0;
                score = a[n - 1] - a[0];
                for(int i = 1; i < n; i++)
                {
                        if(a[i] - k < 0)
                                continue;
```

```
                        high = max(a[i - 1] + k, a[n - 1] - k);
                        low = min(a[0] + k, a[i] - k);
                        score = min(score, high - low);
                }
        }

        cout << score << endl;
        return 0;
}
```

QUESTION 7:
Franky wants to code a problem on strings and you have to help him.
The problem statement is:
Given a string str of length L, find the first non-repeating character in it and return its index. If it does not exist, return -1.
Test case 1:
Input:
codetantra
Output:
0
Test case 2:
Input:
codeincodetantra
Output:
4

ANS:

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
        string str;
        cin >> str;
        int l = str.size();
        int freq[256] = {0};

        int index = l, count = 0;
        for(int i = 0; i < l; i++)
                freq[str[i]]++;

        for(int i = 97; i <= 122; i++)
        {
                if(freq[i] == 1)
                {
                        count++;
                        char c = char(i);
                        for(int j = 0; j < l; j++)
                        {
                                if(c == str[j] && index > j)
```

```
                                        index = j;
                        }
                }
        }
        if(count == 0)
                index = -1;

        cout << index << endl;
        return 0;
}
```

QUESTION 8:
Ramanujam is a mathematician who loves solving problems related to numbers. One day, his friend challenges him with a problem: Given a non-negative integer c, decide whether there are two integers a and b such that a2 + b2 = c.
Sample test case 1:
Input:
5 ------------------- c
Output:
1 ---------------- true
Editorial: 1 * 1 + 2 * 2 = 5
Sample test case 2:
Input:
3
Output:
0 ----------------------false

ANS:

```
#include <iostream>
#include <math.h>

using namespace std;

bool SquareSum(long long int c)
{
        long long int a = 0;
        long long int b = (long)sqrt(c);

        while(a <= b)
        {
                if((a*a) + (b*b) == c)
                        return true;
                else if((a*a) + (b*b) < c)
                        a++;
                else
                        b--;
        }
        return false;
}

int main()
{
        long long int c;
```

```
        cin >> c;

        if(SquareSum(c))
                cout << "1" << endl;
        else
                cout << "0" << endl;
}
```

QUESTION 9:
Yanky is working on a problem where he has an array of integers C of size n, where C[i] is the number of citations a researcher received for their ith paper, return the researcher's h-index.
According to the definition of h-index on Wikipedia: The h-index is defined as the maximum value of h such that the given researcher has published at least h papers that have each been cited at least h times.
Sample test case 1:
Input:
5 ----------------> n
3 0 6 1 5 -------------> C[]
Output:
3

ANS:

```
#include <iostream>

using namespace std;

int main()
{
        int n;
        cin >> n;
        int c[n];
        for(int i = 0; i < n; i++)
                cin >> c[i];

        int l = 0, r = n;
        while(l < r)
        {
                int mid = (l + r + 1) / 2;
                int count = 0;
                for(int i = 0; i < n; i++)
                {
                        if(c[i] >= mid)
                                count++;
                }

                if(count >= mid)
                        l = mid;
                else
                        r = mid - 1;
        }

        cout << l << endl;
```

```
        return 0;
}
```

QUESTION 10:

Mr. Bob is very fond of finding triplets in any given input array/ list A of size n.
Mr. Bob has to find triplets (A[i], A[j], A[k]) such that both below given constraints stands true:
    i != j, i != k, and j != k
    A[i] + A[j] + A[k] == 0.
Note: The solution set must not contain duplicate triplets.
Implement Mr. Bob way of finding triplet's existence by code.
Sample test case 1:
Input:
6 --------> n
-1 0 1 2 -1 -4 -----> Elements in A.
Output:
2
Editorial: Two triplets are (-1,-1,2), and (-1,0,1).
A[0] + A[1] + A[2] = (-1) + 0 + 1 = 0.
A[1] + A[2] + A[4] = 0 + 1 + (-1) = 0.
A[0] + A[3] + A[4] = (-1) + 2 + (-1) = 0.
The distinct triplets are [-1,0,1] and [-1,-1,2].

ANS:

```
#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
        int n;
        cin >> n;
        int a[n];
        for(int i = 0; i < n; i++)
                cin >> a[i];

        sort(a, a + n);
        int count = 0;

        for(int i = 0; i < n - 1; i++)
        {
                int l = i + 1, r = n - 1;
                while(l < r)
                {
                        if(a[i] + a[l] + a[r] < 0)
                                l++;
                        else if(a[i] + a[l] + a[r] > 0)
                                r--;
                        else
                        {
                                count++;
```

```
                                int t1 = l, t2 = r;
                                while(l < r && a[l] == a[t1])
                                        l++;
                                while(l < r && a[r] == a[t2])
                                        r--;
                        }
                }

                while(i + 1 < n && a[i] == a[i + 1])
                        i++;
        }

        cout << count << endl;
        return 0;
}
```

QUESTION 11:
Franky has given a sorted integer array A of size n, two integers k and c, return the k closest integers to c in the array. The result should also be sorted in ascending order.
An integer a is closer to c than an integer b if:
    |a - c| < |b - c|, or
    |a - c| == |b - c| and a < b
Sample test case 1:
Input:
5 ---> n
1 2 3 4 5 ----> A
4 3 ------> k and c respectively
Output:
1 2 3 4

ANS:

```
#include <iostream>

using namespace std;

int main()
{
        int n, k, c;
        cin >> n;
        int a[n];
        for(int i = 0; i < n; i++)
                cin >> a[i];
        cin >> k >> c;

        int l = 0, r = n - k;
        while(l < r)
        {
                int mid = l + (r - l) / 2;
                if(c - a[mid] <= a[mid + k] - c)
                        r = mid;
                else
                        l = mid + 1;
        }
```

```
        for(int i = l; i < l + k; i++)
                cout << a[i] << " ";
        cout << endl;
        return 0;
}
```

QUESTION 12:
Problem Introduction:
The Fibonacci numbers are defined as follows:
F(0) = 0, F(1) = 1, and
F(i) = F(i−1) + F(i−2) for i ≥ 2.
Problem Description:
Given two integers n and m, output F(n) mod m (that is, the remainder of F(n) when divided by m).
Input Format:
The input consists of two integers n and m given on the same line (separated by a space).
Output Format: Output F(n) mod m
Sample test case:
Input:
281621358815590 30524
Output:
11963

ANS:

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

public class CTJ14254 {
        // Matrix multiplication generic form
        public static long[][] matrixMulti(long a[][], long b[][],
int m) {
                long c[][] = new long[a.length][b[0].length];
                for(int i = 0; i < c.length; i++)
                        for(int j = 0; j < b[0].length; j++)
                                for(int k = 0; k < b.length; k++)
                                        c[i][j] = (c[i][j] + (a[i]
[k]*b[k][j]) % m) % m;

                return c;
        }

        // Fast Exponential algorithm implemented on matrix takes
O(logn) time and space
        public static long[][] power(long a[][], long n, int m) {
                if(n == 0) {
                        long x[][] = {
                                {1,0},
                                {0,1}
                        };
                        return x;
                } else if(n == 1) return a;
```

```
                else if(n % 2 == 0) {
                        long c[][] = power(a, n/2, m);
                        return matrixMulti(c, c, m);
                } else {
                        long c[][] = power(a, n/2, m);
                        return matrixMulti(a, matrixMulti(c, c, m),
m);
                }
        }

        public static void main(String args[]) {
                Scanner sc = new Scanner(System.in);
                long n = sc.nextLong();
                int m = sc.nextInt();

                long a[][] = {
                        {1,1},
                        {1,0}
                };
                System.out.println(power(a, n-1, m)[0][0] % m);
        }
}
```

QUESTION 13:

Given an integer array A of size n.

Print the number of triplets chosen from the array that can make triangles if we take them as side lengths of a triangle.

Sample test case 1:

Input:

4 ------------> n

2 2 3 4 ----------> Array A

Output:

3

ANS:

```
#include<bits/stdc++.h>
using namespace std;
int gettriplet(int arr[],int n){
        sort(arr,arr+n);
        int count=0;
        for(int i=n-1;i>=1;i--){
                int l=0,r=i-1;
                while(l<r){
                        if(arr[l]+arr[r]>arr[i]){
                                count+=r-l;
                                r--;
                        }
                        else{
                                l++;
                        }
                }
        }
        return count;

}
```

```
int main(){
        int n;
        cin>>n;
        int arr[n];
        for(int i=0;i<n;i++){
                cin>>arr[i];
        }
        int result=gettriplet(arr,n);
        cout<<result;
        return 0;

}
```

QUESTION 14:
Find the median of the matrix given a row-wise sorted matrix of size RxC where R and C are always odd.
Input format:
    The first line contains two space-separated integers R and C which represent the number of rows and columns of the matrix
    The next lines follow the elements of the matrix with given size R*C.
Output format:
    A single line contains an integer which represents the median in the given matrix.
Sample test case:
Input:
3 3 ---------> R and C
1 3 5 -----> Matrix values entered R x C.
2 6 9
3 6 9
Output:
5

ANS:

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
        int n,m;
        cin>>n>>m;
        int mn= INT_MAX;
        int mx=0;

        int temp;
        vector<vector<int>>nums;
        for(int i=0;i<n;i++)
        {
                vector<int> v;
                for(int j=0;j<m;j++)
                {
                        cin>>temp;
                        v.push_back(temp);
                        mn=min(mn,temp);
```

```
                        mx=max(mx,temp);

                }
                nums.push_back(v);

        }
        int a=(n*m)+1;
        a/=2;
        int mid;
        while(mn<mx)
        {
                mid=mn+(mx-mn)/2;
                int count=0;
                for(int i=0;i<n;i++)
                {

count+=upper_bound(nums[i].begin(),nums[i].end(),mid)-
nums[i].begin();

                }
                if(count<a)
                {
                        mn=mid+1;
                }
else
{
        mx=mid;
}
        }cout<<mn;
        return 0;
}
```

TEST 1:

QUESTION 1:
Given a number n and k. Remove k digits from n , such that you are left with the minimum possible number.
0<n<1018
Example : 503016 and 3, the answer is 1.
503016 and 4 the answer is 0
503016 and 2 answer is 16

```
#include <iostream>
using namespace std;

long MAX = 100000000000000002;

long solve(int idx, int n, long curr, int k, int digits[]) {
        if(k == 0) {
```

```
                while(idx < n) {
                        curr = (curr * 10) + digits[idx];
                        idx++;
                }
                return curr;
        }
        if(idx == n) {
                return MAX;
        }
        long take = solve(idx + 1, n, (curr * 10) + digits[idx], k,
digits);
        long skip = solve(idx + 1, n, curr, k - 1, digits);
        if(take < skip) {
                return take;
        }
        return skip;
}

int main() {
        long n;
        int k;
        cin >> n >> k;
        int digits[20];
        long temp = n;
        int idx = 0;
        while(temp > 0) {
                digits[idx++] = temp % 10;
                temp /= 10;
        }
        int l = 0, r = idx - 1;
        while(l < r) {
                int temp2 = digits[l];
                digits[l] = digits[r];
                digits[r] = temp2;
                l++;
                r--;
        }
        long min = solve(0, idx, 0, k, digits);
        cout << min << "\n";
        return 0;
}
```

QUESTION 2
You are given an array of integers.
In one operation, you can decrease the value of any one element by 2.
Compute the minimum number of operations required to make all the elements in the array distinct.
Example 1 : 1, 3, 2, 1 . Ans is 1.
Example 2: 1, 1, 2 ,2 ,2 . Ans is 4
1->-1, 2->0, 2->0->-2.

ANS:

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
        int n,i,op=0,flag=0;
        cin>>n;
        int a[n];
        for(i=0;i<n;i++)
        {
                cin>>a[i];
        }
        sort(a,a+n);
L1:for(i=1;i<n;i++)
        {
                if(a[i]==a[i-1])
                {
                        a[i]=a[i]-2;
                        op++;
                }
        }
        sort(a,a+n);
        for(i=1;i<n;i++)
        {
                if(a[i]==a[i-1])
                {
                        goto L1;

                }
        }
cout<<op;
return 0;
}
```

QUESTION 3:

F(3208)=13

F(3209)=17

F(3210)=23

F(n)=(119*F(n-1)-131*F(n-2)+1341*F(n-3)-132)%m.

Given m and n, Find the value of F(n);

3210<n<1018 and 10< m<108

Execution Results

ANS:

```
def matm(a,b,h):
        m = len(a)
        p = len(b)
        q = len(b[0])
        c = []
        for i in range(m):
                w = [0]*q
                c.append(w)
```

```
        for i in range(m):
                for j in range(q):
                        for k in range(p):
                                c[i][j]+=a[i][k]*b[k][j]
                        c[i][j]%=h
        return c

def pow_m(a,n,m):
        raw = len(a)
        col = len(a[0])
        if n==0:
                c = []
                for i in range(raw):
                        r=[]
                        for j in range(col):
                                if (i==j):
                                        r.append(1)
                                else:
                                        r.append(0)
                        c.append(r)

                return c
        elif n==1:
                return a
        else:
                temp = pow_m(a,n//2,m)
                ans = matm(temp,temp,m)
                if(n%2):
                        ans = matm(ans,a,m)
                return ans



mat = [[119,-131,1341,-132],[1,0,0,0],[0,1,0,0],[0,0,0,1]]
ini = [[23],[17],[13],[1]]
m,n = map(int,input().split())
c = pow_m(mat,n-3210,m)
c = matm(c,ini,m)
print(c[0][0])
# print(23*119 - 131*17 +1341*13 -132)
```

Practice 2:

QUESTION 1:
Franky is working on binary tree rooted at root, a node N in the tree is named good if in the path from root to N there are no nodes with a value greater than N.

Your task is to find and print the number of good nodes in the binary tree.
Sample test case 1:
Input:
3 1 4 3 null 1 5 -1 // Here -1 is the sentinel value
Output:
4

ANS:

```
#include<bits/stdc++.h>
using namespace std;

class Node{
        public:
        int val;
        Node* left;
        Node* right;

        Node(int v){
                val=v;
                left=NULL;
                right=NULL;
        }
};


int fun(Node* root,stack<int>st){
        if(root==NULL) return 0;

        int c=0;

        if(st.empty() || st.top()<=root->val){
                st.push(root->val);
                c++;
        }

        int l=0,r=0;
        l=fun(root->left,st);
        //cout<<root->val;
        r=fun(root->right,st);
        //if(!st.empty())st.pop();
        // if(st.size() && st.top()<root->val) return l+r+1;
        return l+r+c;

}
int main(){
        vector<string>in;
        string n;
        while(cin>>n){
                if(n.compare("-1")==0) break;
                        in.push_back(n);
        }
```

```
        queue<Node*>q;
        Node* root=new Node(stoi(in[0]));
        q.push(root);
        int i=1;
        while(!q.empty() && i<in.size()){
                Node* temp=q.front();
                q.pop();
                if(in[i].compare("null")!=0){
                        temp->left=new Node(stoi(in[i]));
                        q.push(temp->left);
                }
                i++;
                if(i>=in.size()) break;
                if(in[i].compare("null")!=0){
                        temp->right=new Node(stoi(in[i]));
                        q.push(temp->right);
                }
                i++;
        }
        //cout<<root->val;
        stack<int>st;
        int t=fun(root,st);
        cout<<t;
        return 0;

}
```

QUESTION 2:
Given a binary tree in which each node element contains a number, Find the maximum possible path sum from one leaf node to another leaf node.
Note: Here Leaf node is a node which is connected to exactly one different node.
Input format:
A single line contains space-separated tree nodes. Here, the -1 value denotes the NULL child.
Output format:
Print the maximum possible path sum from one leaf node to another leaf node.

ANS:

```
#include<bits/stdc++.h>
using namespace std;

struct Node{
        int val;
        Node *left, *right;

        Node(){}
        Node(int val){
                this->val = val;
                this->left = NULL, this->right = NULL;
```

```
        }

};


Node* build(string str){
        if(str.length() == 0) return new Node();

        vector<string> inp;
        stringstream ss(str);

        for(string s ; ss >> s;){
                inp.push_back(s);
        }

        Node* root = new Node(stoi(inp[0]));
        queue<Node*> q;
        q.push(root);
        int ind = 1;
        while(!q.empty() and ind < inp.size()){
                Node* tp = q.front(); q.pop();
                string left = inp[ind++];
                if(left != "-1"){
                        tp -> left = new Node(stoi(left));
                        q.push(tp->left);
                }
                if(ind == inp.size()) break;
                string right = inp[ind++];
                if(right != "-1"){
                        tp->right = new Node(stoi(right));
                        q.push(tp->right);
                }
        }
        return root;
}

int dfs(Node* root, int& ans){
        if(root == NULL) return 0;
        int left = dfs(root -> left, ans);
        int right = dfs(root -> right, ans);
        int pick = root->val + left + right;
        ans = max(ans, pick);
        return root -> val + max(left, right);
}

void print(Node* root){
        queue<Node*> q;
        q.push(root);
        while(!q.empty()){
                int sz = q.size();
                for(int i = 0 ; i < sz; i++){
                        Node* tp = q.front(); q.pop();
                        cout << tp->val << " ";
                        if(tp->left) q.push(tp->left);
```

```
                        if(tp->right) q.push(tp->right);
                }
                cout << endl ;
        }
}

int main()
{
        string s;
        getline(cin, s);
        Node* root = build(s);
        int ans = INT_MIN;
        dfs(root, ans);
        cout << ans;

        return 0;
}
```

QUESTION 3:
User has given a N node complete binary tree and user has to check that inputted complete binary tree is a valid binary search tree or not.
You have to complete the function ValidBinarySearchTree with following parameters :
Root : Reference to the root of Binary search tree
Function returns:
As return type is bool you just have to return true for valid and false for invalid BST
Constraints to be followed :
1 < = N < = 103
-103 < = Node values < = 103
Sample test case 1 is :
Input :
3 (N no of nodes to be entered in a BST)
2 3 1 (N data values to be entered level wise same as complete binary tree (Separated with spaces))
Output :
2 NO (Data value at root and status (status is YES if it is valid BST and NO if it is not valid BST)

ANS:

```
#include<bits/stdc++.h>
using namespace std;

class node{
public:
        int data;
        node* left;
        node* right;
        node(int d){
                data = d;
                left = NULL;
```

```
                right = NULL;
        }
};

node* buildTree(){
        int d;
        cin>>d;
        d--;
        int val;
        cin>>val;
        node * root = new node(val);
        queue<node*>q;
        q.push(root);
        while(d){
                node* front = q.front();
                q.pop();
                if(d){
                        cin>>val;
                        front->left = new node(val);
                        q.push(front->left);
                        d--;
                }else{
                        break;
                }

                if(d){
                        cin>>val;
                        front->right = new node(val);
                        q.push(front->right);
                        d--;
                }else{
                        break;
                }
        }
        return root;
}

bool validBST(node* root){
        if(!root || (!root->left && !root->right)) return true;
        bool isBSTL = validBST(root->left);
        bool isBSTR = validBST(root->right);

        int maxLeft = INT_MIN;
        node* temp;
        if(root->left) {
                temp = root->left;
                while(temp->right) temp = temp->right;
                maxLeft = temp->data;
        }

        int minRight = INT_MAX;
        if(root->right){
                temp = root->right;
                while(temp->left) temp = temp->left;
```

```
                minRight = temp->data;
        }
        if(root->data <= maxLeft || root->data >= minRight) return
false;
        if(!isBSTL || !isBSTR) return false;
        if(root->left && root->data < root->left->data) return
false;
        if(root->right && root->data > root->right->data) return
false;

        return true;
}

void printTree(node* root){
        if(!root) return;
        cout<<root->data<<" ";
        printTree(root->left);
        printTree(root->right);
}

int main(){
        node* root = buildTree();
        // printTree(root);
        cout<<root->data<<" ";
        if(validBST(root)) cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
}
```

QUESTION 4:
Yanky and Franky has given the root of a binary search tree (BST), where the values of exactly two nodes of the tree were swapped by mistake.
They have to recover the tree without changing its structure.
Test case 1:
Input:
1 3 null null 2 -1 ----------> -1 is sentinel value, data input is in level order
Output:
3 1 2 ------------> Here we are priniting only the values at each level (levelwise) avoid printing null.
Editorial: 3 cannot be a left child of 1 because 3 > 1. Swapping 1 and 3 makes the BST valid.

ANS:

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define MAX 1000000002

struct Node{
        ll val;
        Node* left;
        Node* right;
```

```
        Node(ll val)
        {
                this->val=val;
                this->left=NULL;
                this->right=NULL;
        }
};

class solution{
        public:
        Node* buildTree(vector<ll>&arr)
        {
                if(arr.size()==0 || arr[0]==MAX) return NULL;
                Node* root=new Node(arr[0]);
                ll n=arr.size();
                queue<Node*>q;
                q.push(root);
        ll i=1;
        while(i<n && !q.empty()){
                Node* par=q.front();
                q.pop();
                Node* lt=NULL;
                Node* rt=NULL;
                if(arr[i]!=MAX)
                {
                        lt=new Node(arr[i]);
                        q.push(lt);
                        par->left=lt;
                }
                i++;
                if(i==n) break;
                if(arr[i]!=MAX)
                {
                        rt= new Node(arr[i]);
                        q.push(rt);
                        par->right=rt;
                }
                i++;
        }
        return root;
        }

        void traverse(Node *root)
        {
                if(!root)
                {
                        cout<<endl;
                }

                queue<Node*> q;
                q.push(root);
                while(!q.empty())
                {
                        Node* par=q.front();
```

```
                        q.pop();
                        cout<<par->val<<" ";
                        if(par->left) q.push(par->left);
                        if(par->right) q.push(par->right);
                }


        }

        vector<Node*> n;
        vector<ll> v;
        void inorder(Node* root)
        {
                if(root==NULL) return;
                inorder(root->left);
                v.push_back(root->val);
                n.push_back(root);
                inorder(root->right);
                return;
        }
        void recover(Node* root)
        {
                inorder(root);
                sort(v.begin(),v.end());
                for(int i=0;i<v.size();i++)
                {
                    n[i]->val=v[i];
                }
                return ;
        }
};

int main()
{
        string temp;
        vector<ll> arr;
        while(1)
        {
                cin>>temp;
                if(temp=="-1") break;
                else if(temp=="null")
                {
                        arr.push_back(MAX);
                }
                else{
                        arr.push_back(stoi(temp));
                }
        }
                solution sol;
                Node *root=sol.buildTree(arr);
                sol.recover(root);
                sol.traverse(root);
```

```
}
```

QUESTION 5:
Yanky has given an integer array A and two integers lower and upper, return the number of range sums that lie in [lower, upper] inclusive.
Range sum S(i, j) is defined as the sum of the elements in A between indices i and j inclusive, where i <= j.
Test case 1:
Input:
3 ------->n
-2 5 -1 ----------> A
-2 ----------> lower
2 ----------> upper
Output:
3

ANS:

```
#include<bits/stdc++.h>

using namespace std;

class SegmentTree{
private:
        int n;
        vector<int> arr;
public:
        SegmentTree(int size): n(size), arr(2 * size){
                arr[0] = 0;
                for(int i = 0; i < n; i++)
                        arr[i + n] = 1;
                for(int i = n - 1; i > 0; i--)
                        arr[i] = arr[2 * i] + arr[2 * i + 1];
        }

        void addValue(int i, int val){
                i += n;
                arr[i] += val;
                while(i > 1)
                {
                        if(i % 2)
                                arr[i / 2] = arr[i - 1] + arr[i];
                        else
                                arr[i / 2] = arr[i] + arr[i + 1];
                        i >>= 1;
                }
        }

        int getSum(int left, int right){
                int count = 0;
                left += n;
                right += n;
```

```
                while(left <= right)
                {
                        if(left % 2)
                                count += arr[left++];
                        if(right % 2 == 0)
                                count += arr[right--];
                        left >>= 1;
                        right >>= 1;
                }
                return count;
        }
};

int countRangeSum(int n, int nums[], int lower, int upper){
        vector<long long> sum(n + 1, 0);
        for(int i = 0; i < n; i++)
                sum[i + 1] = sum[i] + nums[i];

        vector<long long> a(sum.begin(), sum.end());
        sort(a.begin(), a.end());
        SegmentTree tree(a.size());
        int count = 0;
        for(const auto &x : sum)
        {
                int i = lower_bound(a.begin(), a.end(), x) -
a.begin();
                int lowerIndex = lower_bound(a.begin(), a.end(), x +
lower) - a.begin();
                int upperIndex = upper_bound(a.begin(), a.end(), x +
upper) - a.begin() - 1;
                tree.addValue(i, -1);
                count += tree.getSum(lowerIndex, upperIndex);
        }
        return count;
}

int main(){
        int n, lower, upper, range;
        cin >> n;
        int a[n];
        for(int i = 0; i < n; i++)
                cin >> a[i];
        cin >> lower >> upper;
        cout << countRangeSum(n, a, lower, upper) << endl;
}
```

QUESTION 6:
Reconstruction of Binary tree using inorder and postorder traversal's:
Separate arrays namely Post and Ino of size N are given where Post is the preorder traversal of a binary tree and Ino is the in order traversal of the same tree.
You have to re construct the binary tree and return its Root

reference(pointer) to the main module.
You have to write complete code with function BuildTree with following parameters given :
integer Ino[N] : Array representing inorder traversal.
integer Post[N] : Array representing preorder traversal.
0 and N-1 as Start and End indices of the array.
Function returns :
After constructing the binary tree return the Root reference (pointer)
Sample Test Case :
Input:
5 (First line input is N that is no of nodes)
9 3 15 20 7 (Second and third line contains in order traversal)
9 15 7 20 3(Third and third line contains postorder traversal)
Output :
3 9 20 15 7 (Preorder traversal to check the solution)

```
package q14190;
import java.io.*;
import java.util.*;
import java.lang.*;
public class CTJ14190 {
	static class treenode{
		int data;
		treenode left;
		treenode right;
		treenode(int data){
			this.data=data;
			this.left=null;
			this.right=null;
		}
	}
public static void main(String[] args){
	Scanner sc=new Scanner(System.in);
	//print the size of array
	int n=sc.nextInt();
	//print the array size
	int postorder[]=new int[n];
	int inorder[]=new int[n];
	for(int i=0; i<n; i++){
		inorder[i]=sc.nextInt();
	}
	for(int i=0; i<n; i++){
		postorder[i]=sc.nextInt();
	}
//	bulidtree b=new bulidtree();
	treenode newnode= bulidtreey(postorder,inorder);
	preorder(newnode);
}

	/*static class treenode{
		int data;
		treenode left;
		treenode right;
		treenode(int data){
```

```
                        this.data=data;
                        this.left=null;
                        this.right=null;
                }
        }*/
        public static treenode bulidtreey(int[] postorder, int[]
inorder){
                if(inorder==null || postorder==null ||
inorder.length !=postorder.length)return null;
                HashMap<Integer, Integer> map=new
HashMap<Integer,Integer>();
                for(int i=0; i<inorder.length; i++){
                        map.put(inorder[i],i);
                }
                return
bulidtreex(postorder,0,postorder.length-1,inorder,0,inorder.length-1
,map);
        }

        public static treenode bulidtreex(int[] postorder,int
poststart,int postend,int[] inorder,int instart,int
inend,HashMap<Integer,Integer>map){
                if(poststart>postend || instart>inend)return null;
                treenode root=new treenode(postorder[postend]);
                int inroot=map.get(postorder[postend]);
                int numsleft=inroot-instart;


root.left=bulidtreex(postorder,poststart,poststart+numsleft-1,inorde
r,instart,inroot-1,map);

root.right=bulidtreex(postorder,poststart+numsleft,postend-1,inorder
,inroot+1,inend,map);
                return root;
        }
        public static void preorder(treenode root){
                if(root==null){return;}
                System.out.print(root.data + " ");
                preorder(root.left);
                preorder(root.right);
        }

}
```

QUESTION 7:
Range sum query-II:
Franky is given an integer array A and he has to handle multiple queries of the following types:
    Update the value of an element in A.
    Calculate the sum of the elements of A between indices l, and r inclusive where l <= r.
During Implementation you have to define the class IntArray:
    Use argument constructor to Initializes the object with the integer array A.

Define update(index, val) function which updates the value of A[index] to passed value val.
Define rangeSum(l, r) function which returns the sum of the elements of A between indices l and r inclusive i.e. A[l] + A[l + 1] +.....+A[r].
For example, when A = [0,5,2,5,4,3,1,6,3], and (l, r) is (3, 8) then the answer to the range sum query for the sub-array A[3 … 8] = [5,4,3,1,6,3] is 22.
Input:
First line of input contains is the size of the array i.e. n.
Second line contains elements in the array (Space separated).
Third line contains number of queries to be performed i.e. m.
Next lines is the query where u index val represents update value val at given index, and s l r represents find the range sum of the given range (l, r).
Output:
Print the result of all range sum queries before or after update queries.

ANS:

```
#include<bits/stdc++.h>
using namespace std;

void buildTree(int*arr,int* st,int s,int e,int i){
        if(s == e){
                st[i] = arr[s];
                return;
        }

        int mid = s + (e-s)/2;
        buildTree(arr,st,s,mid,2*i+1);
        buildTree(arr,st,mid+1,e,2*i+2);
        st[i] = st[2*i+1] + st[2*i+2];
}

long int rangeSum(int*arr,int*st,int s,int e,int l,int r,int i){
        if(s > r || l > e) return 0;
        if(l <= s and r >= e) return  st[i];

        int mid = s + (e-s)/2;
        return rangeSum(arr,st,s,mid,l,r,2*i+1) +
rangeSum(arr,st,mid+1,e,l,r,2*i+2);
}

void update(int* st,int s,int e,int idx,int val,int i){
        if(idx < s || idx > e) return;
        if(s == e) {
                st[i] = val;
                return;
        }

        int mid = s + (e-s)/2;
        update(st,s,mid,idx,val,2*i+1);
        update(st,mid+1,e,idx,val,2*i+2);
```

```
        st[i] = st[2*i+1] + st[2*i+2];
}


int main(){
        int n;
        cin>>n;
        int arr[n];
        vector<long int>v;
        for(int i= 0; i<n; i++)cin>>arr[i];
        int q;
        cin>>q;
        int st[4*n];
        buildTree(arr,st,0,n-1,0);
        while(q--){
                char c; int l,r;
                cin>>c>>l>>r;
                if(c == 's'){
                        long int s = rangeSum(arr,st,0,n-1,l,r,0);
                        v.push_back(s);
                }
                else update(st,0,n-1,l,r,0);
        }
        for(auto x : v)cout<<x<<" ";
}
```

QUESTION 8:
Franky has to write complete code where he has to construct the tree rooted at reference root.
As a problem given a binary tree he has to find its minimum depth.
Editorial:
The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.
Note: A leaf is a node with no children.
Sample test case 1:
Input:
3 9 20 null null 15 7 -1 ---------> Enter elements in the tree levelwise and sentinel value is -1.
Output:
2 --------> minimum path depth.

ANS:

```
#include<bits/stdc++.h>
using namespace std;

//vector<int>v;
struct node
{
        int val;
        node *left;
        node *right;
        node(int val)
        {
```

```
                this->val=val;
                this->left=NULL;
                this->right=NULL;
        }
};



node* createTree(vector<string>&input)
{

node* root = new node(stoi(input[0]));
queue<node*>q;
q.push(root);
int i=1;
int n=input.size();
while(!q.empty())
{
        node *curr=q.front();
        q.pop();
        if(i<n)
        {
                if(input[i]=="null")
                {
                        curr->left=NULL;
                }
                else
                {
                        curr->left = new node(stoi(input[i]));
                        q.push(curr->left);
                }
        }
        i++;
        if(i<n)
        {
                if(input[i]!="null")
                {
                        curr->right=new node(stoi(input[i]));
                        q.push(curr->right);
                }
        }
        i++;
}
return root;

}




int min_depth(node *root)
{
        if(!root)return 0;
```

```
        if(!root->left&&!root->right)return 1;
        int l = INT_MAX;
        int r = INT_MAX;
        if(root->left)
        {
                l=min_depth(root->left);

        }
        if(root->right)
        {
                r =min_depth(root->right);
        }


        return 1+min(l,r);

}


int main()
{
        vector<string>input;
        string prev="a";
        while(prev!="-1")
        {
                string tmp;
                cin>>tmp;
                prev=tmp;
                input.push_back(tmp);

        }
        input.pop_back();
        node *root = createTree(input);
        int m_depth = min_depth(root);
        cout<<m_depth;
        return 0;

}
```

QUESTION 9:
You are given a 0-indexed integer array P of size n, where P[i] represents the number of stones in the ith pile, and an integer t.
You should apply the following operation exactly t times:
Choose any pile P[i] and remove floor(P[i]/2) stones from it.
Notice that you can apply the operation on the same pile more than once.
Your task is to print the minimum possible total number of stones remaining after applying the t operations.
Note: floor(x) is the greatest integer that is smaller than or equal to x (i.e., rounds x down).
Sample tes case 1:
Input:
3---> n
5 4 9 ----------> P[]

2-----> t
Output:
12

ANS:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
        int n;
        cin>>n;
        int a;
        vector<int>arr;
        for(int i=0;i<n;i++)
        {
                cin>>a;
                arr.push_back(a);
        }
        int t;
        cin>>t;
        priority_queue<int>pq;
        for(int i:arr)
        {
                pq.push(i);
        }
        while(t--)
        {
                int top1 = pq.top();
                pq.pop();
                int top2;
                top2 = top1-floor(top1/2);
                pq.push(top2);
        }
        int sum=0;
        while(!pq.empty())
        {
                sum = sum+pq.top();
                pq.pop();
        }
        cout<<sum;
        return 0;
}
```

QUESTION 10:
Franky is working on two 0-indexed integer arrays a1 and a2, each of size n, and an integer d. Find the number of pairs (i, j) such that:
 0 <= i < j <= n - 1
and   a1[i] - a1[j] <= a2[i] - a2[j] + d.
Franky has to print the number of pairs that satisfy the conditions.
Test case 1:
Input:
3 ----------> n
3 2 5 -----> a1[]
2 2 1 ------->a2[]

1 ---------> d
Output:
3

ANS:

```
import java.io.*;
import java.lang.*;
import java.util.*;
public class CTJ16672 {
        public static void main(String args[]){
                Scanner sc=new Scanner(System.in);

                //print the size of array
                int n=sc.nextInt();
                int a1[]=new int[n];
                int a2[]=new int[n];
                for(int i=0; i<n; i++){
                        a1[i]=sc.nextInt();

                }
                for(int i=0; i<n; i++){
                        a2[i]=sc.nextInt();
                }
                int count=0;
                //print the value of d
int d=sc.nextInt();
for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
                if((a1[i]-a1[j])<=(a2[i]-a2[j]+d)){
                        count++;
                }
        }
}
System.out.println(count);
        }

}
```

QUESTION 11:
Range minimum query:
Given an array A[0 … n − 1] of n objects taken from a totally ordered set, such as integers, the range minimum query RMQA(l,r) =arg min A[k] (with 0 ≤ l ≤ k ≤ r ≤ n − 1) returns the index of the minimal element in the specified sub-array A[l … r].
For example, when A = [0,5,2,5,4,3,1,6,3], then the answer to the range minimum query for the sub-array A[3 … 8] = [2,5,4,3,1,6] is 7, as A[7] = 1.
Constraints:
    1 <= n <= 1000
    -100000 <= A[i] <= 100000
    Segment tree for implementation
    If query is invalid print -1.
Input:

First line of input contains is the size of the array i.e. n.
Second line contains elements in the array (Space separated).
Third line is the query representing range l and r.

Output:

Print the index of the min value as required.

ANS:

```
#include<bits/stdc++.h>
using namespace std;

void buildTree(int* st,int* arr,int s,int e,int i){
        if(s == e){
                st[i] = s;
                return;
        }
        int mid = s + (e-s)/2;
        buildTree(st,arr,s,mid,2*i+1);
        buildTree(st,arr,mid+1,e,2*i+2);
        if(arr[st[2*i+1]] < arr[st[2*i+2]]) st[i] = st[2*i+1];
        else st[i] = st[2*i+2];
}

int rmq(int*arr,int* st,int s,int e,int l ,int r ,int i){
        if(l > e || r < s) return -1;
        if(l <= s and r >= e) return st[i];

        int mid = s + (e-s)/2;
        int leftSe = rmq(arr,st,s,mid,l,r,2*i+1);
        int rightSe = rmq(arr,st,mid+1,e,l,r,2*i+2);

        if(leftSe == -1 and rightSe == -1) return -1;
        else if(leftSe == -1) return rightSe;
        else if(rightSe == -1) return leftSe;

        if(arr[st[leftSe]] < arr[st[rightSe]])return leftSe;
    return rightSe;
}

int main(){
        int n;
        cin>>n;
        int arr[n];
        for(int i = 0; i< n; i++) cin>>arr[i];
        int l,r;
        cin>>l>>r;
        if(l < 0 or r >= n){
         cout<<"-1"<<endl;
         return 0;
        }
        int st[4*n];
        buildTree(st,arr,0,n-1,0);
        cout<<rmq(arr,st,0,n-1,l,r,0)<<endl;
}
```

QUESTION 12:
Franky has given the root of a binary tree, find the maximum value v for which there exist different nodes a and b where v = |a.val - b.val| and a is an ancestor of b.
Note: A node a is an ancestor of b if either: any child of a is equal to b or any child of a is an ancestor of b.
Sample test case 1:
Input:
8 3 10 1 6 null 14 null null 4 7 13 -1 // Here -1 is the sentinal value
Output:
7
Editorial: We have various ancestor-node differences, some of which are given below :

ANS:

```
#include<iostream>
#include<vector>
#include<climits>
#include<queue>
#include<algorithm>

using namespace std;

struct Node
{
        int val;
        Node* left;
        Node* right;

        Node(int x):val(x), left(NULL),right(NULL){}
};
Node* build(vector<string>& v )
{
        if(v.empty())
          return NULL;

         int i=0;
         Node* root = new Node(stoi(v[i++]));
         queue<Node*> q;
         q.push(root);
         while(i<v.size())
         {
                Node* node= q.front();
                q.pop();
                if(v[i]!="null")
                {
                        node->left = new Node(stoi(v[i]));
                        q.push(node->left);
                }
                i++;
```

```
                if(i<v.size()&&v[i]!="null")
                {
                        node->right= new Node(stoi(v[i]));
                        q.push(node->right);
                }
                i++;
         }

        return root;
}
int maximum(Node* root, int maxval,int minval)
{

        //considr all the path and find the maximum and the min in
the path .record all the leaf to the root min and max;
        if(root==NULL)
                return abs(maxval-minval);

                maxval = max(root->val, maxval);
                minval = min(root->val, minval);

                int left = maximum(root->left,maxval,minval);
                int right= maximum(root->right,maxval,minval);

                return max(left,right);
}

int main()
{
        vector<string> v;
        while(1)
        {
                string s;
                cin>>s;
                if(s=="-1")
                 break;

                v.push_back(s);

        }


        Node* root = build(v);
         if(root==NULL)
           cout<<0;


        int maxval= INT_MIN;
        int minval =INT_MAX;
    int ans=maximum(root,maxval,minval);
        cout<<ans;

        return 0;
}
```

QUESTION 13:
Franky has to write complete code where he has to construct the tree rooted at reference root.
As a problem he has to check whether it is a mirror of itself (i.e., symmetric around its center).
Sample test case 1:
Input:
1 2 2 3 4 4 3 -1 ---------> Enter elements in the tree levelwise and sentinel value is -1.
Output:
1 --------> 1 represents true

ANS:

```
#include<bits/stdc++.h>
using namespace std;

class node{
public:
        int data;
        node* left;
        node* right;
        node(int d){
                data = d;
                left = NULL;
                right = NULL;
        }
};

node* buildTree(){
        string str;
        cin>>str;
        node* root;
        if(str == "null") return NULL;
        else root = new node(stoi(str));
        queue<node*> q;
        q.push(root);
        while(1){
                node* front = q.front();
                q.pop();
                cin>>str;
                if(str == "-1")break;
                if(str != "null"){
                        front->left = new node(stoi(str));
                        q.push(front->left);
                }
                else front->left = NULL;
                cin>>str;
                if(str == "-1") break;
                else if(str != "null"){
                        front->right = new node(stoi(str));
                        q.push(front->right);
                }
                else front->right = NULL;
```

```
        }
        return root;
}

bool isSym(node* a,node* b){
        if(!a and !b) return true;
        if((!a and b) || (a and !b)) return false;

        bool mirror1 = isSym(a->left,b->right);
        bool mirror2 = isSym(a->right,b->left);

        if(mirror1 && mirror2 && a->data == b->data) return true;

        return false;
}

int main(){
        node* root = buildTree();
        if((!root->left && root->right) || (root->left && !root-
>right)){
                cout<<"0"<<endl;

        }


        else cout<<isSym(root->left,root->right)<<endl;


}
```

QUESTION 14:
You are given an integer array G of size n denoting the number of gifts in various piles. Every second, you do the following:
Choose the pile with the maximum number of gifts.
    If there is more than one pile with the maximum number of gifts, choose any.
    Leave behind the floor of the square root of the number of gifts in the pile. Take the rest of the gifts.
Your task is to print the number of gifts remaining after k seconds.
Sample test case 1:
Input:
5 -----> n
25 64 9 4 100 -------> G[]
4 --------> k
Output:
29

ANS:

```
#include<bits/stdc++.h>

using namespace std;
```

```
int main(){
        long long n, k, gift = 0;
        cin >> n;
        vector<long long> g;
        for(int i = 0; i < n; i++)
        {
                cin >> k;
                g.push_back(k);
        }
        cin >> k;

        priority_queue<long long> pq(g.begin(), g.end());
        while(k--)
        {
                long long a = pq.top();
                pq.pop();
                pq.push(sqrt(a));
        }

        while(pq.size())
        {
                gift += pq.top();
                pq.pop();
        }
        cout << gift << endl;
}
```

QUESTION 15:
There are n software interns standing in a line. Each software intern is assigned a unique rating value.
You have to form a team of 3 software interns amongst them under the following constraints:
    Choose 3 software intern with index (i, j, k) with rating (r[i], r[j], r[k]).
    A team is valid if: (r[i] < r[j] < r[k]) or (r[i] > r[j] > r[k]) where 0 <= i < j < k < n.
You have to find the number of teams you can form given the conditions.
Note: software interns can be part of multiple teams.
Sample test case 1:
Input:
5 ------------> n
2 5 3 4 1 ----------> ratings in r[]
Output:
3 -----------> Count of teams

ANS:

```
#include <bits/stdc++.h>
using namespace std;

int solve(vector<int>&nums,int n){
        if(nums.size()<3){
                return 0;
        }
```

```
        int ans=0;
        for(int i=1;i<n-1;i++){

                int LL=0;
                int LG=0;
                int RL=0;
                int RG=0;
                for(int j=0;j<i;j++){
                        if(nums[j]<nums[i]) LL++;
                        if(nums[j]>nums[i]) LG++;
                }
                for(int j=i+1;j<n;j++){
                        if(nums[j]<nums[i]) RL++;
                        if(nums[j]>nums[i]) RG++;
                }
                ans+=(LL*RG +LG*RL);
        }
        return ans;


}

int main(){
        int n;
        cin>>n;
        vector<int>nums;
        int temp;
        for(int i=0;i<n;i++){
                cin>>temp;
                nums.push_back(temp);

        }

        cout<<solve(nums,n);

}
```

QUESTION 16:
Range sum query:
Franky is given an integer array A and he has to handle multiple queries of the following types:
    Calculate the sum of the elements of A between indices l, and r inclusive where l <= r.
During Implementation you have to define the class IntArray:
    Use argument constructor to Initializes the object with the integer array A.
    Define rangeSum(l, r) function which returns the sum of the elements of A between indices l and r inclusive i.e. A[l] + A[l + 1] +.....+A[r].
For example, when A = [0,5,2,5,4,3,1,6,3], and (l, r) is (3, 8) then the answer to the range sum query for the sub-array A[3 … 8] =

[5,4,3,1,6,3] is 22.
Input:
    First line of input contains is the size of the array i.e. n.
    Second line contains elements in the array (Space separated).
    Third line is the query representing range l and r.
Output:
    Print the range sum.

ANS:

```
#include <bits/stdc++.h>
using namespace std;

class IntArray{
        vector<int>seg;
        public:
                IntArray(int n){
                        seg.resize(4*n+1);
                }

                void build(int ind,int low,int
high,vector<int>&nums){
                        if(low==high){
                                seg[ind]=nums[low];
                                return;
                        }
                        int mid=low+(high-low)/2;
                        build(2*ind+1,low,mid,nums);
                        build(2*ind+2,mid+1,high,nums);
                        seg[ind]=seg[2*ind+1]+seg[2*ind+2];
                }
                int query(int ind,int low,int high,int l, int r){
                        if(low>=l && high<=r) return seg[ind];
                        if(r<low|| high<l) return 0;
                        int mid=low+(high-low)/2;
                        int left=query(2*ind+1,low,mid,l,r);
                        int right=query(2*ind+2,mid+1,high,l,r);
                        return left+right;
                }

};

int main(){
        int n;
        cin>>n;
        int temp;
        vector<int>nums;
        for(int i=0;i<n;i++){
                cin>>temp;
                nums.push_back(temp);
        }
        int l,r;
        cin>>l>>r;
        if(l<0 || r>n-1 || l>r){
                cout<<-1;
```

```
        }
        else{
                IntArray sol(n);
                sol.build(0,0,n-1,nums);
                cout<<sol.query(0,0,n-1,l,r);
        }

}
```

QUESTION 17:
Range frequency query:
Franky is given an integer array A and he has to handle range frequency query of the following type:
Find the frequency of a given value val in a given subarray.
    The frequency of a value val in a subarray is the number of occurrences of that value in the subarray between indices l, and r inclusive where l <= r.
During Implementation you have to define the class RangeFrequency:
    Use argument constructor to Initializes the object with the integer array A.
    Define rangeQuery(l, r, val) function which returns the frequency of the given value val in A between indices l and r inclusive i.e. A[l ---- r].
Note: Subarray is a contiguous sequence of elements within an array
For example, when A = [0,5,2,5,4,3,1,6,3], and (l, r, 5) is (0, 3, 5) then the answer to the range frequency query for the sub-array A[0 … 3] = [0, 5, 2, 5] is 2.

Input:
First line of input contains is the size of the array i.e. n.
    Second line contains elements in the array (Space separated).
    Third line contains number of queries to be performed i.e. m.
    Next lines is the query where (l, r, val) represents left, right, and value attributes of the query.
Output:
    Print the result of all range sum queries before or after update queries.

ANS:

```
#include <bits/stdc++.h>
using namespace std;

class RangeFrequency{
        vector<unordered_map<int,int>>seg;
        public:
        RangeFrequency(int n){
                seg.resize(4*n+1);
        }
        void build(int ind,int low,int high,vector<int>&nums){
                if(low==high){
                        seg[ind][nums[low]]++;
                        return;
                }
                int mid=low+(high-low)/2;
```

```
                build(ind*2+1,low,mid,nums);
                build(ind*2+2,mid+1,high,nums);
                for(auto it:seg[2*ind+1]){
                        seg[ind][it.first] += it.second;
                }
                for(auto it:seg[2*ind+2]){
                        seg[ind][it.first]+=it.second;
                }
                return;
        }
        int query(int ind ,int low,int high,int l,int r,int val){
                if(r<low|| high<l) return 0;
                if(low>=l && high<=r){
                        //int ans=0;
                        return seg[ind][val];
                }
                int mid=low+(high-low)/2;
                int left=query(2*ind+1,low,mid,l,r,val);
                int right=query(2*ind+2,mid+1,high,l,r,val);
                return left+right;
        }
};

int main(){
        int n;
        cin>>n;
        int temp;
        vector<int>nums;
        for(int i=0;i<n;i++){
                cin>>temp;
                nums.push_back(temp);
        }
        int m;
        cin>>m;
        RangeFrequency sol(n);
        sol.build(0,0,n-1,nums);
        vector<int>ans;
        while(m){
                m--;
                int l,r,val;
                cin>>l>>r>>val;
                if(l<0||r>n-1||r<l||l>n-1||r<0){
                        ans.push_back(-1);
                }
                else{
                        ans.push_back(sol.query(0,0,n-1,l,r,val));
                }
        }
        for(auto it:ans){
                cout<<it<<" ";
        }
}
```

QUESTION 18:

Given an input stream of N integers. The task is to insert these numbers into a new stream and find the median of the stream formed by each insertion of X to the new stream.
Input format:
The first line contains an integer N, representing the number of integers in the stream.
The second line contains n space-separated array X[ ] of integers.
Output format:
Print the median element in the stream for each insertion.
Sample test case:
Input:
4
5 15 1 3
Output:
5
10
5
4

ANS:

```
package q14279;
import java.io.*;
import java.util.*;
import java.lang.*;
public class CTJ14279 {
public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        //print the array size
        int n=sc.nextInt();
        //print the array
        int x[]=new int[n];
        for(int i=0; i<n; i++){
                x[i]=sc.nextInt();
        }
        for(int i=0; i<n; i++){
                int[] b=new int[i+1];
                for(int j=0; j<=i; j++){
                        b[j]=x[j];
                }
                Arrays.sort(b);
                if(b.length %2!=0){System.out.println(b[b.length/
2]);}
                else{
                        if((b[(b.length-1)/2] + b[b.length/2])%2==0)
{
                                System.out.println((b[(b.length-1)/
2] + b[b.length/2])/2);
                        }
                        else{
                        System.out.println((double)(b[(b.length-1)/
2] + b[(b.length)/2])/2);}}
        }

}
```

```
}
```

QUESTION 19:
Reconstruction of Binary tree inorder and preorder traversal's:
Separate arrays namely Pre and Ino of size N are given where Pre is the preorder traversal of a binary tree and Ino is the in order traversal of the same tree.
You have to re construct the binary tree and return its Root reference(pointer) to the main module.
You have to write complete code with function BuildTree with following parameters given :
integer Ino[N] : Array representing inorder traversal.
integer Pre[N] : Array representing preorder traversal.
0 and N-1 as Start and End indices of the array.
Function returns :
After constructing the binary tree return the Root reference (pointer)
Sample Test Case :
Input:
5 (First line input is N that is no of nodes)
9 3 15 20 7 (Second and third line contains in order traversal)
3 9 20 15 7 (Third and third line contains preorder traversal)
Output :
9 15 7 20 3 (Postorder traversal to check the solution)

ANS:

```
package q14189;
import java.io.*;
import java.util.*;
import java.lang.*;
public class CTJ14189 {
static class treenode{
        int data;
        treenode left;
        treenode right;
        treenode(int data){
                this.data=data;
                this.left=null;
                this.right=null;
        }

}

public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        //print the size of array
        int n=sc.nextInt();
        //print the array
        int[] preorder=new int[n];
        int[] inorder=new int[n];
        for(int i=0; i<n; i++){
                inorder[i]=sc.nextInt();
        }
```

```
        for(int i=0; i<n; i++){
                preorder[i]=sc.nextInt();
        }
        treenode newnode=buildtree(preorder,inorder);
        postorder(newnode);

}


public static treenode buildtree(int[] preorder,int[] inorder){
        HashMap<Integer,Integer> map=new HashMap<Integer,Integer>();
        for(int i=0; i<inorder.length; i++){
                map.put(inorder[i],i);
        }
        return
buildtreex(preorder,0,preorder.length-1,inorder,0,inorder.length-1,m
ap);
}

public static treenode buildtreex(int[] preorder,int prestart,int
preend,int[] inorder,int instart,int
inend,HashMap<Integer,Integer>map){
        if(prestart>preend || instart>inend)return null;
        treenode root=new treenode(preorder[prestart]);
        int inroot=map.get(root.data);
        int numsleft=inroot-instart;

root.left=buildtreex(preorder,prestart+1,prestart+numsleft,inorder,i
nstart,inroot-1,map);

root.right=buildtreex(preorder,prestart+numsleft+1,preend,inorder,in
root+1,inend,map);
        return root;
}
public static void postorder(treenode root){
        if(root==null){return;}
        postorder(root.left);
        postorder(root.right);
        System.out.print(root.data + " ");
}
}
```

QUESTION 20:
Yanky want to solve a unique problem on binary search tree's.
Problem is:
Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST.
According to the concept LCA on wiki: “The lowest common ancestor is defined between two nodes a and b as the lowest node in BST that has both a and b as descendants.
Note: We allow a node to be a descendant of itself
Tree taken in below 1 and 2 test cases is:
Sample test case 1:
Input:

6 2 8 0 4 7 9 null null 3 5 -1 ----------> Here data is in levelorder and -1 is the sentinel value
2 8 ----------> a, and b
Output:
6
Editorial: The LCA of nodes 2 and 8 is 6.

ANS:

```
#include<bits/stdc++.h>
using namespace std;

class node{
public:
        int data;
        node* left;
        node* right;
        node(int d){
                data = d;
                left = NULL;
                right = NULL;
        }
};

node* buildTree(){
        string str;
        cin>>str;
        node* root;
        if(str == "null") return NULL;
        else root = new node(stoi(str));

        queue<node*>q;
        q.push(root);
        while(1){
                node* front = q.front();
                q.pop();
                cin>>str;
                if(str == "-1") break;
                else if(str != "null"){
                        front->left = new node(stoi(str));
                        q.push(front->left);
                }
                else front->left = NULL;
                cin>>str;
                if(str == "-1") break;
                else if(str != "null"){
                        front->right = new node(stoi(str));
                        q.push(front->right);
                }
                else front->right = NULL;
        }
        return root;
}

node* lca(node* root,int a,int b){
```

```
        if(!root) return NULL;

        if(root->data == a || root->data == b) return root;
        if(a < root->data && b < root->data) return lca(root-
>left,a,b);
        else if(a > root->data && b > root-> data) return lca(root-
>right,a,b);
                node* leftlca = lca(root->left,a,b);
                node* rightlca = lca(root->right,a,b);

                if(leftlca and rightlca) return root;
                else if(!rightlca) return leftlca;
                 return rightlca;

}

int main(){
        node* root = buildTree();
        int a,b;
        cin>>a>>b;
        cout<<lca(root,a,b)->data<<endl;
}
```

QUESTION 21:
Franky did his home work on binary tree, now Professor Bordo gave him a problem based on concept of infinite binary tree to solve.
Editorial:
In an infinite binary tree where every node has two children, the nodes are labelled in row order.
In the odd numbered rows (ie., the first, third, fifth,...), the labelling is left to right, while in the even numbered rows (second, fourth, sixth,...), the labelling is right to left.
In the problem Franky has given the label L of a node in this tree, return the labels in the path from the root of the tree to the node with that label L.
Sample test case 1:
Input:
14 --------> L
Output:
1 3 4 14 ------> Path to 14 as required

ANS:

```
#include<bits/stdc++.h>

using namespace std;

int main(){
        int label;
        cin >> label;

        vector<int> path;
        path.push_back(label);
```

```
        int d = log2(label), m;
        while(d)
        {
                m = pow(2, d) + pow(2, d + 1) - 1 - label;
                path.push_back(m / 2);
                label = m / 2;
                d--;
        }
        reverse(path.begin(), path.end());
        for(int i = 0; i < path.size(); i++)
                cout << path[i] << " ";
        cout << endl;
}
```

QUESTION 22:
Yanky is designing a module where an integer array A of size n is available and he has to find the number of reverse pairs in the array.
A pair (i, j) is a reverse pair when it satisfies both conditions:
    0 <= i < j < n
    A[i] > 2*A[j]
Sample test case 1:
Input:
5
1 3 2 3 1
Output:
2

ANS:

```
#include <bits/stdc++.h>
using namespace std;
#define  ll long long

int merge(vector<ll>&nums,int low,int mid,int high){
        vector<ll> temp;
        int left=low;
        int right=mid+1;
        int pair=0;
        int j=mid+1;
        for(int i=low;i<=mid;i++){
                while(j<=high && nums[i]>2*nums[j]){
                        j++;
                }
                pair+=j-(mid+1);
        }
        while(left<=mid && right<=high){
                if(nums[left]>nums[right]){
                        temp.push_back(nums[right]);
                        right++;
                }
                else{
                        temp.push_back(nums[left]);
                        left++;
```

```
                }
        }
        while(left<=mid){
                temp.push_back(nums[left]);
                left++;
        }
        while(right<=high){
                temp.push_back(nums[right]);
                right++;
        }
        for(int i=0;i<temp.size();i++){
                nums[low+i]=temp[i];
        }
        return pair;
}
int mergesort(vector<ll>&nums,int low,int high){
        if(low>=high) return 0;
        int mid=low+(high-low)/2;
        int a=mergesort(nums,low,mid);
        int b=mergesort(nums,mid+1,high);
        int c=merge(nums,low,mid,high);

        return a+b+c;
}

int main(){
        int n;
        cin>>n;
        ll temp;
        vector<ll>nums;
        for(int i=0;i<n;i++){
                cin>>temp;
                nums.push_back(temp);
        }
        cout<< mergesort(nums,0,n-1);
}
```

QUESTION 23:
Yanky want to solve a unique problem on binary search tree with n nodes.
Problem is:
Given the root of a binary search tree, and an integer k, Yanky has to find the kth smallest value (1-indexed) of all the values of the nodes in the tree.
Tree taken in below 1 and 2 test cases is:
Sample test case 1:
Input:
6 2 8 0 4 7 9 null null 3 5 -1 ----------> Here data is in levelorder and -1 is the sentinel value
1 ----------> k
Output:
0

ANS:

```
package q14201;
import java.io.*;
import java.util.*;
import java.lang.*;
class node{
        node left;
        node right;
        int val;
        public node(int val){this.val=val;}
}
public class CTJ14201 {
        public static void main(String[] args)throws IOException{
                BufferedReader bf=new BufferedReader(new
InputStreamReader(System.in));
                String a[]=bf.readLine().split(" ");
                int[] arr=new int[a.length-1];
                Scanner sc=new Scanner(System.in);
                int k=sc.nextInt();
                for(int i=0; i<arr.length; i++){
                        if(a[i].equals("null")){
                                arr[i]=-1;
                        }
                        else{
                                arr[i]=Integer.parseInt(a[i]);
                        }
                }
                node root=new node(arr[0]);
                buildtree(root,arr);
                List<Integer> in=new ArrayList<>();
                inorder(root,in);
                System.out.println(in.get(k-1));
        }

        public static void inorder(node root, List<Integer> in){
                if(root==null)return;
                inorder(root.left,in);
                in.add(root.val);
                inorder(root.right,in);
        }

        public static void buildtree(node root,int[] arr){
                Queue<node> q=new LinkedList<>();
                q.offer(root);
                int i=1;
                while(i<arr.length){
                        int num=arr[i];
                        node curr=q.poll();
                        if(num!=-1){
                                curr.left=new node(num);
                                q.offer(curr.left);
                        }
                        i++;
                        if(i<arr.length){
```

```
                                num=arr[i];
                                if(num !=-1){
                                        curr.right=new node(num);
                                        q.offer(curr.right);
                                }
                        }
                        i++;
                }

        }

/*      static int idx=0;
        static class node{
                int data;
                node left;
                node right;
                node(int data){
                        this.data=data;
                        this.left=null;
                        this.right=null;
                }
        }
        static class binarytree{
        //      static int idx=-1;
                public static node buildtree(int nodes[]){
                //      static int idx=-1;
                        if(idx>nodes.length) return null;
                        idx++;


                        if(nodes[idx]==-1){return null;}
                        node newnode=new node(nodes[idx]);
                        newnode.left=buildtree(nodes);
                        newnode.right=buildtree(nodes);
                        return newnode;
                        //else{break;}
                }
        }
        public static void inorder(node root, int[] b,int i){
                if(root==null){return;}
                inorder(root.left,b,i);
                b[i++]=root.data;
                inorder(root.right,b,i);
        }
        public static void main(String[] args)throws IOException{
        //      Scanner sc=new Scanner(System.in);
                int[] a;
                int i=0;
                while(true){
                        a[i]=sc.nextInt();
                        if(a[i]==-1){break;}
                }
                InputStreamReader in=new
InputStreamReader(System.in);
```

```
                BufferedReader bf=new BufferedReader(in);
                String ip=bf.readLine();
                String[] a=ip.split(" ");
                int[] ar=new int[a.length-1];
                for(int i=0; i<ar.length; i++){
                        ar[i]=Integer.parseInt(a[i]);
                }

                Scanner sc=new Scanner(System.in);
                //print the value of k
                int k=sc.nextInt();
                int[] b=new int[ar.length];
                //binarytree tree=new binarytree();
                node root=binarytree.buildtree(ar);
                inorder(root,b,0);
                System.out.println(b[k]);
        }
}*/
}
```

QUESTION 24:
Yanky is working with binary tree rooted at rootand an integer D. A pair of two different leaf nodes of a binary tree is said to be good if the length of the shortest path between them is less than or equal to D.
Task is to print the number of good leaf node pairs in the tree.
Sample test case 1:
Input:
1 2 3 null 4 -1 // here -1 is sentinel value
3 // its distance D
Output:
1
Editorial: The leaf nodes of the tree are 3 and 4 and the length of the shortest path between them is 3. This is the only good pair.

ANS:

```
#include<bits/stdc++.h>
using namespace std;
class node{
public:
        int data;
        node* left;
        node* right;
        node(int d){
                data = d;
                left = NULL;
                right = NULL;
        }
};

node* buildTree(){
        string str;
        cin>>str;
        node* root;
```

```
        queue<node*>q;
        if(str == "null") return NULL;
        else{
                root = new node(stoi(str));
                q.push(root);
        }

        while(1){
                node* front = q.front();
                q.pop();
                cin>>str;
                if(str == "-1") break;
                if(str != "null"){
                        front->left = new node(stoi(str));
                        q.push(front->left);
                }
                cin>>str;
                if(str == "-1") break;
                if(str != "null"){
                        front->right = new node(stoi(str));
                        q.push(front->right);
                }
        }
        return root;
}

void printTree(node* root){
        if(!root) return;
        cout<<root->data<<" ";
        printTree(root->left);
        printTree(root->right);
}

map<int,int> findGoodPairs(node* root,int &cnt,int d){
        map<int,int>mp;
        if(!root) return mp;
        if(!root->left && !root->right){
                if(mp.count(0)) mp[1]++;
                else mp[1] = 1;
                return mp;
        }
        map<int,int> leftSe = findGoodPairs(root->left,cnt,d);
        map<int,int> rightSe = findGoodPairs(root->right,cnt,d);
        for(auto l : leftSe){
                for(int i = 0; i <= d - l.first; i++){
                        if(rightSe.count(i)) cnt += (l.second *
rightSe[i]);
                }
                mp[l.first + 1] = l.second;
        }
        for(auto r : rightSe){
                if(mp.count(r.first + 1)){
                        mp[r.first + 1] += r.second;
                }else{
```

```
                        mp[r.first + 1 ] = r.second;
                }
        }
        return mp;
}

int main(){
        node* root = buildTree();
        // printTree(root);
        int d;
        cin>>d;
        int cnt = 0;
        findGoodPairs(root,cnt,d);
        cout<<cnt<<endl;

}
```

QUESTION 25:
You have to complete the function TargetSumPath , The parameters available are :
Root pointer referencing to a binary tree and an integer Target.
TargetSumPath will return true if the tree has any path from root to leaf such that adding up all the values along the path equals to the Target value given.
Constraints to be followed:
1 < = N (No of nodes) < = 2000.
0<= Data value <= 1000
0<= Target <= 1000
Sample test case :
5 (N i.e no of nodes in the binary tree)
1 2 3 4 null 5 6 7 (Data entered in the binary tree in level order) (null means no node inserted)
14 (Target value)
Output :
true

ANS:

```
#include<bits/stdc++.h>
using namespace std;
class node{
public:
        int data;
        node* right;
        node* left;
        node(int d){
                data = d;
                right = NULL;
                left = NULL;
        }
};

node* buildTree(){
        int n;
        cin>>n;
```

```
        string d;
        cin>>d;
        node* root;
        queue<node*>q;
        if(d == "null") return NULL;
        else {
                root = new node(stoi(d));
                q.push(root);
        }
        n--;
        while(n){
                cin>>d;
                node* front = q.front();
                q.pop();
                n--;
                if(d != "null"){
                        front->left = new node(stoi(d));
                        q.push(front->left);
                }
                else front->left = NULL;

                if(!n)break;
                cin>>d;
                n--;
                if(d != "null"){
                        front->right = new node(stoi(d));
                        q.push(front->right);
                }
                else front ->right = NULL;
        }
        return root;
}
bool findTr(node* root,int target,int val){
        if(!root) return false;
        if(!root->left and !root->right){
                val += root->data;
                if(val == target) return true;
                else return false;
        }
        val += root->data;
        return findTr(root->left,target,val) || findTr(root-
>right,target,val);
}
int main(){
        node* root = buildTree();
        int target;
        cin>>target;
        int val = 0;
        if(findTr(root,target,val)){
                cout<<"true"<<endl;
        }
        else{
                cout<<"false"<<endl;
        }
```

}

TEST 2:

QUESTION 1:
You are given an integer array G of size n denoting the number of gifts in various piles. Every second, you do the following:
Choose the pile with the maximum number of gifts.
    If there is more than one pile with the maximum number of gifts, choose any.
    Leave behind the floor of the square root of the number of gifts in the pile. Take the rest of the gifts.
Your task is to print the number of gifts remaining after k seconds.
Sample test case 1:
Input:
5 -----> n
25 64 9 4 100 -------> G[]
4 --------> k
Output:
29

ANS:
```
#include<bits/stdc++.h>

using namespace std;

int main(){
        long long n, k, gift = 0;
        cin >> n;
        vector<long long> g;
        for(int i = 0; i < n; i++)
        {
                cin >> k;
                g.push_back(k);
        }
        cin >> k;

        priority_queue<long long> pq(g.begin(), g.end());
        while(k--)
        {
                long long a = pq.top();
                pq.pop();
                pq.push(sqrt(a));
        }
        while(pq.size())
        {
                gift += pq.top();
```

```
                pq.pop();
        }
        cout << gift << endl;
}
```

QUESTION 2:
You are given a complete binary tree with n nodes, with each node having a integer value, (positive/negative).
You know that in any tree, there is a unique path between any two nodes. Weight of a path is defined as the sum of the values of the nodes in the path, including both the end points.
Compute the weight of the path with maximum weight.
Note that, if all the nodes values are negative the answer is 0 and if one of the node value is positive, then the ans will be positive.
Input: First line n
Second line n integers
Output one integer.
Note that a complete binary tree can be stored in an array !

ANS:

```
#include<bits/stdc++.h>
using namespace std;
struct Node{
        int data;
        Node* right;
        Node* left;
        Node(int val){
                data=val;
                right=left=NULL;
        }
};
Node* build(int arr[],int i,int n){
        Node* root=NULL;
        if(i<n){
                root=new Node(arr[i]);
                root->left=build(arr,2*i+1,n);
                root->right=build(arr,2*i+2,n);
        }
        return root;
}
/*void inorder(Node* root){
        if(root==NULL) return;
        inorder(root->left);
        cout<<root->data<<"\n";
        inorder(root->right);
}*/
int solve(Node* root,int& sum){
        if(root==NULL) return 0;
        int left_sum=max(0,solve(root->left,sum));
        int right_sum=max(0,solve(root->right,sum));
        sum=max(root->data+left_sum+right_sum,sum);
        return root->data+max(left_sum,right_sum);
```

```
}
int main(){
        int n;
        cin>>n;
        int arr[n];
        for(int i=0;i<n;i++){
                cin>>arr[i];
        }
        int flag=0;

        for(int i=0;i<n;i++){    //for all elements that are below 0
                if(arr[i]<0){
                        flag-=1;
                }
        }
        if(flag==-(n)){
                cout<<0;
                return 0;
        }


        Node* root=build(arr,0,n);
        int sum=INT_MIN;
        //inorder(root);
        solve(root,sum);
        cout<<sum;
        return 0;
}
```

QUESTION 3:
You are given an array of integers . Implement an algorithm to compute the number of triplets such that
I<J<K
3A[I]+5>A[J]>5A[K] +3
Input: First line an integer n
Second line n integers

ANS:

```
#include<bits/stdc++.h>
using namespace std;

int main() {
        int n;
        cin >> n;
        int nums[n];
        for(int i = 0; i< n; i++)       cin >> nums[i];

        if(n < 3) {
                cout<<"0";
                return 0;
        }
        int ans = 0;
        for(int i = 1; i < n-1; i++) {
```

```
                int left = 0, right = 0;
                for(int j = 0; j < i; j++) {
                        if(3*nums[j]+5 > nums[i])
                                left++;
                }
                for(int j = i + 1; j < n; j++) {
                        if(nums[i] > 5*nums[j]+3)
                                right++;
                }
                ans = ans + (left*right);
        }
        cout<<ans;
}
```