**Ans to the Quest. No:1**

#include<bits/stdc++.h>

using namespace std;

const int N = 1e5+7;

int nums[N];

void merge(int l, int mid, int r){

// if(l>=r){return;}

int n1 = mid-l+1;

int n2 = r-mid;

int left[n1 + 1],right[n2+1];

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

left[j] = nums[i];

}//left sorted array

for(int i=mid+1,j=0;i<=r;i++,j++){

right[j] = nums[i];

}//right sorted array

left[n1] = INT\_MAX;//infinity

right[n2] = INT\_MAX;//infinity

int lp = 0,rp=0;

for(int i=l;i<=r ;i++){

if(left[lp] <= right[rp]){

nums[i] = left[lp];

lp++;}

else{

nums[i] = right[rp];

rp++;

}

}//merge the two sorted arrays using two pointers

}

void reverse(int start, int end){

while(start<end){

int temp = nums[start];

nums[start] = nums[end];

nums[end] = temp;

start++;

end--;

}

}

void mergesort(int l,int r){

if(l>=r){return;}

int mid = (l+r)/2;

mergesort(l,mid);

mergesort(mid+1,r);

merge(l,mid,r);

}

int32\_t main(){

int n;

cin>>n;

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

cin>>nums[i];

}

mergesort(0,n-1);

reverse(0,n-1);

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

cout<<nums[i]<<" ";

}

return 0;

}

**Ans to the Quest. No: 2**

#include<bits/stdc++.h>

using namespace std;

const int N = 1e5+7;

int searchk(vector<int>v,int l,int r,int k){

if(l>r){return -1;}

int mid = (l+r)/2;

if(v[mid]==k){

return mid;

}

if(v[mid]>k){return searchk(v,l,mid-1,k);}

else{return searchk(v,mid+1,r,k);}

}

int32\_t main(){

int n;

cin>>n;

vector<int> v(n);

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

cin>>v[i];

}

int k;cin>>k;

if(searchk(v,0,n-1,k)==-1){

cout<<"Not Found";

}else{

cout<<searchk(v,0,n-1,k);

};

return 0;

}

**Ans to the Quest. No:3**

#include<bits/stdc++.h>

using namespace std;

const int N = 1e5+7;

int searchkFirst(vector<int>v,int l,int r,int k){

if(l>r){return -1;}

int mid = (l+r)/2;

if(v[mid]==k){

if (mid == 0 || v[mid - 1] != k) {

return mid;

} else {

return searchkFirst(v, l, mid - 1, k);

}

}

if(v[mid]>k){return searchkFirst(v,l,mid-1,k);}

else{return searchkFirst(v,mid+1,r,k);}

}

int searchkLast(vector<int>v,int l,int r,int k){

if(l>r){return -1;}

int mid = (l+r)/2;

if(v[mid]==k){

if (mid == v.size()-1 || v[mid + 1] != k) {

return mid;

} else {

return searchkLast(v, mid+1, r, k);

}

}

if(v[mid]>k){return searchkLast(v,l,mid-1,k);}

else{return searchkLast(v,mid+1,r,k);}

}

int32\_t main(){

int n;

cin>>n;

vector<int> v(n);

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

cin>>v[i];

}

int k;cin>>k;

if(searchkFirst(v,0,n-1,k)!=searchkLast(v,0,n-1,k)){

cout<<"YES";

}else{

cout<<"NO";

};

return 0;

}

**Ans to the Quest No :4**

1. If we calculate frist loop its complexity O(log2n), and second loop complexity is O(n/5). So, total complexity is O((n/5)\*log2n) approximately T.C = nlog2n or nlogn
2. T.C = O(sqrt(n)), as i = sqrt(n)
3. First loop complexity is O(log2n) and the second loop complexity is O(sqrt(n)). So, the total complexity is O( sqrt(n)\*log2n)
4. T.C = O(sqrt(n))

**Ans to the Quest. No:5**

#include<bits/stdc++.h>

using namespace std;

vector<int>v1(1000),v2(1000);

vector<int> mergeTwoVector(vector<int>&v1,vector<int>&v2){

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

vector<int>v3(v1.size()+v2.size());

int lp = 0,rp=0;

for(int i=0,j=0;i<v1.size(),j<v2.size();){

if(v1[i] >= v2[j]){

v3[k] = v1[i];

i++;}

else{

v3[k] = v2[j];

j++;

}

k++;

}

return v3;

}

int32\_t main(){

int n;

cin>>n;

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

cin>>v1[i];

}

int m;

cin>>m;

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

cin>>v2[i];

}

vector<int>v3= mergeTwoVector(v1,v2);

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

cout<<v3[i]<<" ";

}

return 0;

}