**Week 1**

**Q1) Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input)**

**#include** <bits/stdc++.h>

using **namespace** std;

**int** **main**()

**int** t;

    cin**>>**t;

**while**(t--) {

**int** n, key;

        cin**>>**n;

**int** arr[n];

**for**(**int** i = 0; i < n; i++) {

            cin**>>**arr[i];

        }

        cin**>>**key;

**int** j;

**for**(j = 0; j < n; j++) {

**if**(arr[j] == key) {

                cout**<<**"Present "**<<**j+1**<<**'\n';

**break**;

            }

        }

**if**(j == n) {

            cout**<<**"Not Present "**<<**j**<<**"\n";

        }

    }

}

**OUTPUT**



**Q2) Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(nlogn), where n is the size of input).**

#include <bits/stdc++.h>

using namespace std;

int main() {

int t;

cin>>t;

while(t--) {

int n, key;

cin>>n;

int arr[n];

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

cin>>arr[i];

}

cin>>key;

int l = 0, h = n - 1;

int i = 0, flag = 0;

while(l <= h) {

int mid = (l + h) / 2;

i++;

if(arr[mid] == key)

{

flag = 1;

cout<<"Present "<<i<<endl;

break;

}

else if(arr[mid] > key) h = mid - 1;

else l = mid + 1;

}

if(!flag) cout<<"Present "<<n/2<<endl;

}

return 0;

}

**OUTPUT**



**Q3) Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4],.....,arr[2k] and so on. Once the interval (arr[2k] < key < arr[ 2k+1] ) is found, perform a linear search operation from the index 2k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching):**

**Jump Search**

**Input format:**

**The first line contains number of test cases, T.**

**For each test case, there will be three input lines.**

**First line contains n (the size of array).**

**Second line contains n space-separated integers describing array.**

**Third line contains the key element that need to be searched in the array.**

**Output format:**

**The output will have T number of lines.**

**For each test case, output will be “Present” if the key element is found in the array, otherwise “Not Present”.**

**Also, for each test case output the number of comparisons required to search the key.**

#include <bits/stdc++.h>

using namespace std;

void jump(int arr[], int n, int key) {

int start = 0, comp = 0, flag = 0;

int end = sqrt(n);

while(arr[end] <= key && end < n) {

comp++;

start = end;

end += sqrt(n);

if(end > n-1) end = n;

}

for(int i=start;i<end;i++) {

if(arr[i] == key) {

flag = true;

break;

}

}

if(flag) cout<<"Present "<<comp<<endl;

else cout<<"Not present"<<endl;

}

int main()

{

int n;

cin>>n;

while(n--)

{

int size;

cin>>size;

int arr[size];

for(int i = 0; i < size; i++) cin>>arr[i];

int key;

cin>>key;

jump(arr, size, key);

}

return 0;

}

**OUTPUT**



**Week 2**

**Q1) Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))**

**Input format:**

**The first line contains number of test cases, T.**

**For each test case, there will be three input lines.**

**First line contains n (the size of array).**

**Second line contains space-separated integers describing array.**

**Third line contains the key element that need to be searched in the array.**

**Output format:**

**The output will have T number of lines.**

**For each test case T, output will be the key element and its number of copies in the array if the key element is present in the array otherwise print “Key not present”.**

#include<iostream>

using namespace std;

int main() {

int n,x;

int count=0;

cout<<"enter the size of array:"<<endl;

cin>>n;

int A[n];

cout<<"\n Enter the element of array:"<<endl;

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

{

cin>>A[i];

}

cout<<"\n Enter the element you want to search"<<endl;

cin>>x;

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

{

if(A[i]==x)

{

count++;

}

}

cout<<"The frequency of key element is :"<<count;

return 0;

}

**OUTPUT**



**Q2) Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].**

**Input format:**

**The first line contains number of test cases, T.**

**For each test case, there will be two input lines.**

**First line contains n (the size of array).**

**Second line contains space-separated integers describing array.**

**Output:**

**The output will have T number of lines.**

**For each test case T, print the value of i, j and k, if found else print “No sequence found”.**

#include <bits/stdc++.h>

using namespace std;

int main() {

int t;

cin>>t;

while(t--) {

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

vector<int> v;

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

int k = n - 1;

int j = i + 1;

while(j < n && k > j) {

if(arr[i] + arr[j] == arr[k]) {

v.push\_back(i+1);

v.push\_back(j+1);

v.push\_back(k+1);

break;

}

else if(arr[i] + arr[j] > arr[k]) {

j++;

k = n - 1;

}

else k--; }

}

if(v.empty())

cout<<"No sequence found"<<endl;

else {

for(auto it: v) {

cout<<it<<" ";

}

cout<<endl; }

}

}

**OUTPUT**



**Q3) Given an array of nonnegative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.**

**Input format:**

**The first line contains number of test cases, T.**

**For each test case, there will be three input lines.**

**First line contains n (the size of array).**

**Second line contains space-separated integers describing array. Third line contains the key element.**

**Output format:**

**The output will have T number of lines.**

**For each test case T, output will be the total count i.e., number of times such pair exists.**

#include <bits/stdc++.h>

using namespace std;

int main() {

int t;

cin>>t;

while(t--) {

int n;

cin>>n;

int arr[n];

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

cin>>arr[i];

int key;

cin>>key;

sort(arr, arr + n);

int c = 0, l, h;

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

l = i;

h = n - 1;

while(l < h) {

if(arr[h] - arr[l] == key) {

c++;

h--;

l++;

}

else if(arr[h] - arr[l] > key) h--;

else l++;

}

}

cout<<c<<endl;

}

}

**OUTPUT**



**Week 3**

**Q1) Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts (shifts total number of times the array elements are shifted from their place) required for sorting the array.**

**Input Format:**

**The first line contains number of test cases, T.**

**For each test case, there will be two input lines.**

**First line contains n (the size of array).**

**Second line contains space-separated integers describing array.**

**Output Format:**

**The output will have T number of lines.**

**For each test case T, there will be three output lines.**

**First line will give the sorted array.**

**Second line will give total number of comparisons.**

**Third line will give total number of shift operations required.**

#include<bits/stdc++.h>

using namespace std;

void insertionSort(int a[], int n){

int ctr=0,flag=0;

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

ctr++;

int j;

j=i-1;

int x=a[i];

while (j>-1 && a[j]>x){

ctr++;

a[j+1]=a[j];

flag++;

j--;

}

a[j+1]=x;

}

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

cout<<a[i]<<" ";

}

cout<<endl;

cout<<" Shifts = "<<ctr<<endl;

cout<<"Comparison = "<<flag<<endl;

}

int main(){

int n;

cout<<"enter the no. of test cases"<<endl;

cin>>n;

while(n--)

{

int c;

cout<<"enter the no. of elements in an array"<<endl;

cin>>c;

int a[c];

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

{

cin>>a[i];

}

insertionSort(a,c);

}

}

**OUTPUT**



**Q2) Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.**

**Input Format:**

**The first line contains number of test cases, T.**

**For each test case, there will be two input lines.**

**First line contains n (the size of array).**

**Second line contains space-separated integers describing array.**

**Output Format:**

**The output will have T number of lines.**

**For each test case T, there will be three output lines.**

**First line will give the sorted array.**

**Second line will give total number of comparisons.**

**Third line will give total number of swaps required.**

#include<bits/stdc++.h>

using namespace std;

void selectionSort(int a[], int n){

int ctr=0,flag=0,k;

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

ctr++;

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

flag++;

if(a[i]>a[j])

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

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

cout<<a[i]<<" ";

}

cout<<endl;

cout<<"Swaps = "<<ctr<<endl;

cout<<"Comparison = "<<flag<<endl;

}

int main(){

int n;

cout<<"enter the no. of test cases"<<endl;

cin>>n;

while(n--)

{

int c;

cout<<"enter the no. of elements in an array"<<endl;

cin>>c;

int a[c];

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

{

cin>>a[i];

}

selectionSort(a,c);

}

}

**OUTPUT**



**Q3) Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting)**

**(Time Complexity = O(n log n))**

**Input Format:**

**The first line contains number of test cases, T.**

**For each test case, there will be two input lines.**

**First line contains n (the size of array).**

**Second line contains space-separated integers describing array.**

**Output Format:**

**The output will have T number of lines.**

**For each test case, output will be 'YES' if duplicates are present otherwise ‘NO’.**

#include<bits/stdc++.h>

using namespace std;

void findDuplicate(int a[], int x)

{

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

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

if(a[i]==a[j]){

cout<<"YES"<<endl;

}

break;

}

}

cout<<"NO"<<endl;

}

int main(){

int n;

cin>>n;

while(n--){

int x;

cin>>x;

int a[x];

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

cin>>a[i];

}

findDuplicate(a ,x);

}

}

**OUTPUT**



**Week 4**

**Q1) Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array.**

**Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array.**

**Output Format: The output will have T number of lines. For each test case T, there will be three output lines. First line will give the sorted array. Second line will give total number of comparisons. Third line will give total number of inversions required.**

#include <bits/stdc++.h>

using namespace std;

int c = 0;

void mergeArray(int arr[], int l, int mid, int h) {

int n1 = mid - l + 1;

int n2 = h - mid;

int a[n1], b[n2];

for(int i = 0; i < n1; i++) {

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

}

for(int i = 0; i < n2; i++) {

b[i] = arr[mid + 1 + i];

}

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

while(i < n1 && j < n2) {

c++;

if(a[i] <= b[j]) {

arr[k] = a[i];

i++;

}

else {

arr[k] = b[j];

j++;

}

k++;

}

while(i < n1) {

arr[k] = a[i];

i++; k++;

}

while(j < n2) {

arr[k] = b[j];

j++; k++;

}

}

void mergeSort(int arr[], int l, int h) {

if(l < h) {

int mid = l + (h - l) / 2;

mergeSort(arr, l, mid);

mergeSort(arr, mid + 1, h);

mergeArray(arr, l, mid, h);

}

}

void display(int arr[], int n) {

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main() {

int t;

cin>>t;

while(t--) {

int n;

cin>>n;

int \*arr = new int[n];

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

cin>>arr[i];

}

mergeSort(arr, 0, n - 1);

display(arr, n);

cout<<"comparisons = "<<c<<endl;

delete []arr;

}

}

**OUTPUT**



**Q2) Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another sub array holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array.**

**Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array.**

**Output Format: The output will have T number of lines. For each test case T, there will be three output lines. First line will give the sorted array. Second line will give total number of comparisons. Third line will give total number of swaps required.**

#include <bits/stdc++.h>

using namespace std;

int c = 0, s = 0;

int partition(int arr[], int l, int h) {

int x = (rand() % (l - h)) + l;

if(h != x) {

s++;

swap(arr[x], arr[h]);

}

int pivot = arr[h];

int i = l - 1;

for(int j = l; j <= h - 1; j++) {

if(arr[j] <= pivot) {

i++;

s++;

swap(arr[i], arr[j]);

}

}

s++;

swap(arr[i + 1], arr[h]);

return i + 1;

}

void quickSort(int arr[], int l, int h) {

if(l < h) {

int pivot = partition(arr, l, h);

quickSort(arr, l, pivot - 1);

quickSort(arr, pivot + 1, h);

}

}

void display(int arr[], int n) {

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

cout<<arr[i]<<" ";

}

cout<<endl;

}

int main() {

int t;

cin>>t;

while(t--) {

int n;

cin>>n;

int \*arr = new int[n];

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

cin>>arr[i];

}

quickSort(arr, 0, n - 1);

display(arr, n);

cout<<"swaps = "<<s<<endl;

}

}

**OUTPUT**



**Q3) Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))**

**Input Format: The first line contains number of test cases, T. For each test case, there will be three input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Third line contains K.**

**Output Format: The output will have T number of lines. For each test case, output will be the Kth smallest or largest array element. If no Kth element is present, output should be “not present”.**

#include<bits/stdc++.h>

using namespace std;

int main() {

int t;

cin>>t;

while(t--) {

int n;

cin>>n;

int \*arr = new int[n];

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

cin>>arr[i];

}

int k;

cin>>k;

priority\_queue<int> pq;

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

pq.push(arr[i]);

}

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

pq.push(arr[i]);

if(pq.size() > k) pq.pop();

}

if(pq.empty()) cout<<"Not present"<<endl;

else cout<<pq.top()<<endl;

delete []arr;

}

}

**OUTPUT**



**Week 5**

**Q1) Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it. (Time Complexity = O(n)) (Hint: Use counting sort)**

**Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array.**

**Output: The output will have T number of lines. For each test case, output will be the array element which has maximum occurrences and its total number of occurrences. If no duplicates are present (i.e. all the elements occur only once), output should be “No Duplicates Present”.**

#include<iostream>

using namespace std;

char findmax(char a[], int n)

{

char m=a[0];

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

{

if(a[i]>m)

{

m=a[i];

}

}

return m;

}

void count\_sort(char a[],int n)

{

int i,\*c,j,dup,temp;

char max=findmax(a,n);

int m=int(max);

c = new int[m+1];

for(i=0;i<m+1;i++)

{

c[i]=0;

}

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

{

c[int(a[i])]++;

}

dup=c[i];

for(int i=0;i<m+1;i++)

{

if(c[i]>dup)

{

dup=c[i];

temp=i;

}

}

char ch=(char)temp;

if(dup>1)

{

cout<<ch<<" :"<<dup<<endl;

}

else

{

cout<<"No duplicates found"<<endl;

}

}

int main()

{

int t,n;

cout<<"enter number of test case"<<endl;

cin>>t;

while(t>0)

{

cout<<"enter number of elements in array"<<endl;

cin>>n;

char a[n];

cout<<"enter "<<n<<" number of elements"<<endl;

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

{

cin>>a[i];

}

count\_sort(a,n);

t--;

}

}

**OUTPUT**



**Q2) Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n))**

**Input Format: The first line contains number of test cases, T. For each test case, there will be two input lines. First line contains n (the size of array). Second line contains space-separated integers describing array. Third line contains key**

**Output Format: The output will have T number of lines. For each test case, output will be the elements arr[i] and arr[j] such that arr[i]+arr[j] = key if exist otherwise print 'No Such Elements Exist”.**

#include <bits/stdc++.h>

using namespace std;

int main()

{

int t;

cin >>t;

while(t--)

{

int n,sum,flag=0;

cin >> n;

int ar[n];

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

cin >> ar[i];

cin >> sum;

sort(ar,ar+n);

int a=0,b=n-1;

while(a<b)

{

if(ar[a]+ar[b]>sum)

--b;

else if(ar[a]+ar[b]<sum)

++a;

else

{

cout << ar[a] << "&" << ar[b] << ", ";++a;--b;++flag;

}

}

if(!flag)

cout << "NO Such Pair Exist" << "\n";

}

}

**OUTPUT**



**Q3) You have been given two sorted integer arrays of size m and n. Design an algorithm and implement it using a program to find list of elements which are common to both. (Time Complexity = O(m+n))**

**Input Format: First line contains m (the size of first array). Second line contains m space-separated integers describing first array. Third line contains n (the size of second array). Fourth line contains n space-separated integers describing second array.**

**Output Format: Output will be the list of elements which are common to both.**

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n,m;

cin >> n ;

int ar[n];

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

cin >> ar[i];

cin >> m;

int ar2[m];

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

cin >> ar2[i];

int a=0,b=0;

while(a<n && b<m)

{

if(ar[a] < ar2[b])

++a;

else if(ar[a] > ar2[b])

++b;

else

{

cout << ar[a] << " ";++a;++b;}

}

cout << "\n";

}

**OUTPUT**

