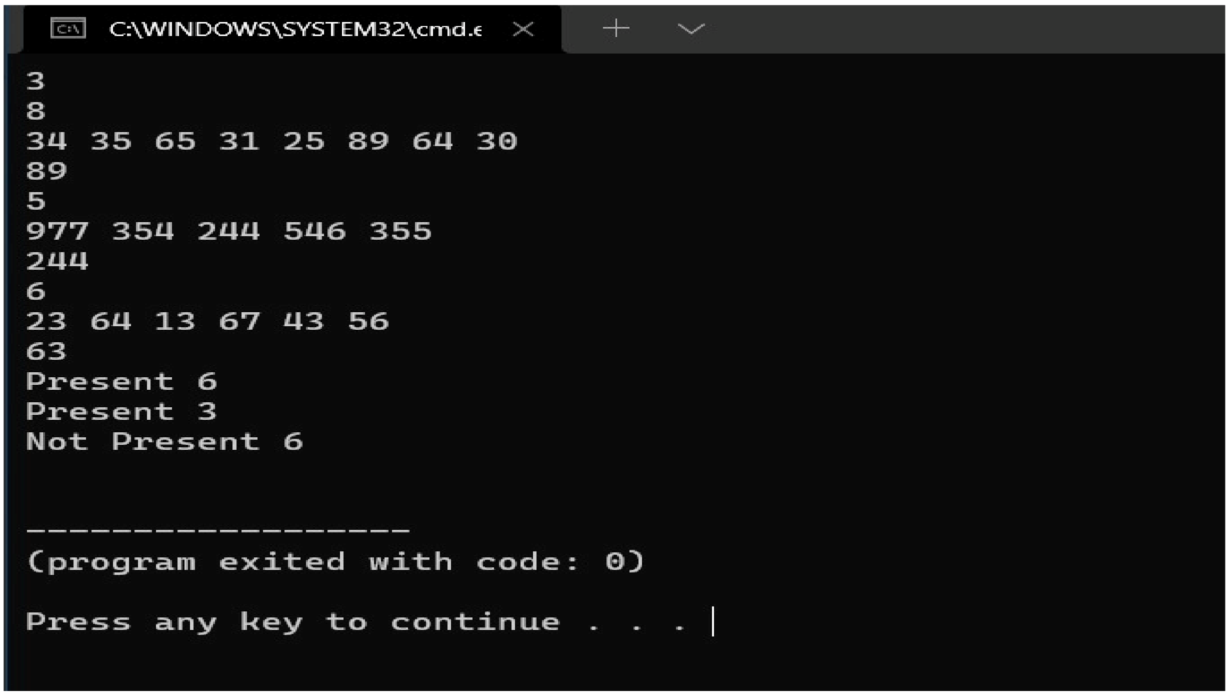
Week 1:

I. 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<iostream>
#include<vector>
using namespace std;

int main(){
	int t;
	cin>>t;
	while(t--){
		int n;
		cin>>n;
		vector<int> ar(n);
		for(int &i:ar) cin>>i;
		int key;
		cin>>key;
		int cnt=0;
		int fl=0;
		for(int i:ar){
			cnt++;
			if(i==key){
			  fl=1;
			  break;


		}
		if(fl){
			cout<<"Present "<<cnt<<"\n";
		}else{
			cout<<"Not Present "<<cnt<<"\n";
		}
	}
	return 0;
}
```

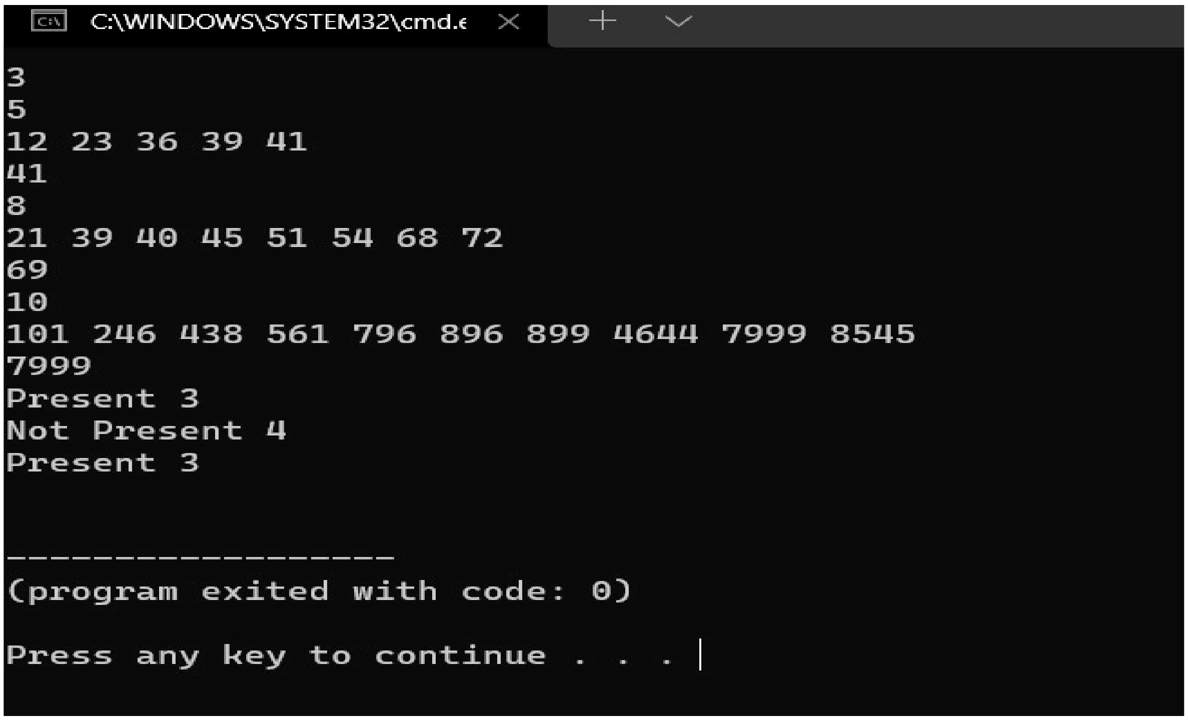


C:\WINDOWS\SYSTEM32\cmd.e

```
3
8
34 35 65 31 25 89 64 30
89
5
977 354 244 546 355
244
6
23 64 13 67 43 56
63
Present 6
Present 3
Not Present 6


------------------
(program exited with code: 0)

Press any key to continue . . .
```

II. 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(logn), where n is the size of input)

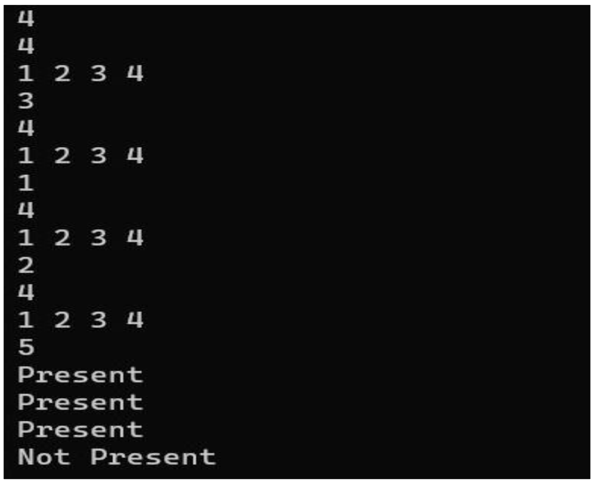
```
#include<iostream>
#include<vector>
using namespace std;

int main(){
        int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                int key;
                cin>>key;
                int cnt=0;
                int l=0;
                int r=n-1;
                while(l<=r){
                        int mid = l + (r-l)/2;
                        cnt++;
                        if(ar[mid]==key){
                                        break;
                        }else if(ar[mid]>key) r=mid-1;
                        else l=mid+1;
                }
                if(l<=r){
                        cout<<"Present "<<cnt<<"\n";
                }else{
                        cout<<"Not Present "<<cnt<<"\n";
                }
        }
        return 0;
}
```


C:\WINDOWS\SYSTEM32\cmd.e

```
3
5
12 23 36 39 41
41
8
21 39 40 45 51 54 68 72
69
10
101 246 438 561 796 896 899 4644 7999 8545
7999
Present 3
Not Present 4
Present 3


------------------
(program exited with code: 0)

Press any key to continue . . .
```

III. 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):

```
#include<bits/stdc++.h>
using namespace std;
int jump(vector<int>  &ar,int key){
                int n=ar.size();
                int m=sqrt(n);
                int i=0;
                while(i<n){
                                if(ar[i]==key) return 1;
                                if(ar[i]>key){
                                        for(int j=max(0,i-m)+1;j<i;j++)  if(ar[j]==key) return 1;
                                        return 0;
                                }
                                if(i+m>=n){
                                        for(;i<n;i++) if(ar[i]==key) return 1;
                                        return 0;
                                }
                                i+=m;
                }
        return 0;
}

int main(){
        int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                int key;
                cin>>key;
                if(jump(ar,key)){
                        cout<<"Present\n";
                }else{
                        cout<<"Not Present\n";
                }
        }
        return 0;
}
```

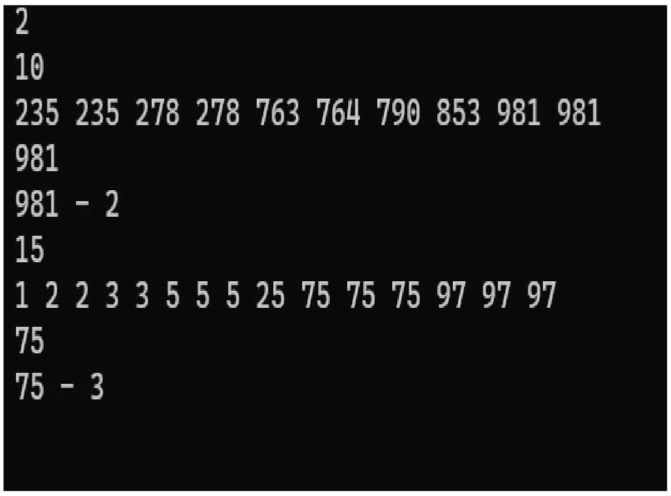


```
4
4
1 2 3 4
3
4
1 2 3 4
1
4
1 2 3 4
2
4
1 2 3 4
5
Present
Present
Present
Not Present
```

Week 2:

I. 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)).

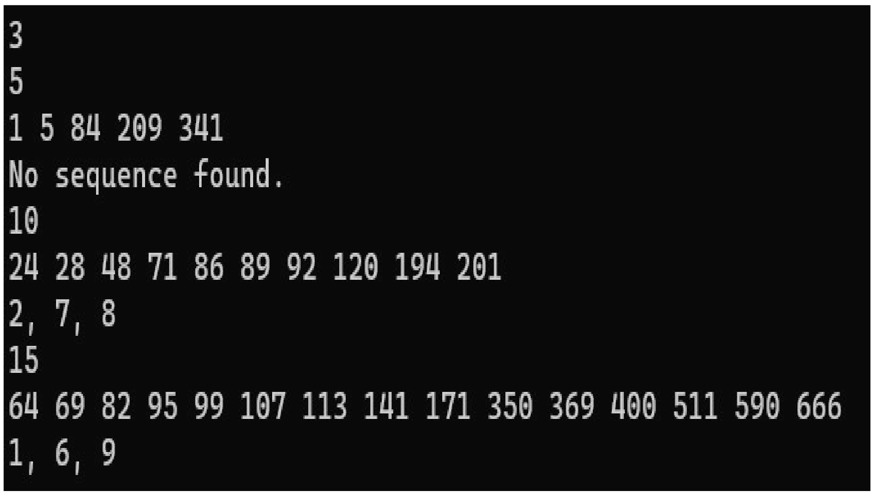
```
#include<bits/stdc++.h>
using namespace std;
int main(){
        int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                int key;
                cin>>key;
                int l=0;
                int r=n-1;
                int f,ls;
                f=ls=-1;
                while(l<=r){
                        int mid = l + (r-l)/2;
                        if(ar[mid]==key){
                                        f=mid;
                                        r=mid-1;
                        }else if(ar[mid]>key) r=mid-1;
                        else l=mid+1;
                }
                l=0;
                r=n-1;
                while(l<=r){
                        int mid = l + (r-l)/2;
                        if(ar[mid]==key){
                                        ls=mid;
                                        l=mid+1;
                        }else if(ar[mid]>key) r=mid-1;
                        else l=mid+1;
                }
                if(f==-1) cout<<"Not present\n";
                else cout<<key<<" - "<<(ls-f+1)<<"\n";    }
return 0;
}
```



```
2
10
235 235 278 278 763 764 790 853 981 981
981
981 - 2
15
1 2 2 3 3 5 5 5 25 75 75 75 97 97 97
75
75 - 3
```

II. 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].

```
#include<bits/stdc++.h>
using namespace std;
int main(){
int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                bool found=0;
                for(int i=0;i<n && !found;i++){
                        for(int j=i+1;j<n && !found;j++){
                                for(int k=j+1;k<n;k++){
                                        if(ar[i] + ar[j] == ar[k]){
                                                cout<<i+1<<", "<<j+1<<", "<<k+1<<"\n";
                                                found=1;
                                                break;
                                        }
                                }
                        }
                }
                if(!found) cout<<"No sequence found.\n";
        }
return 0;
}
```

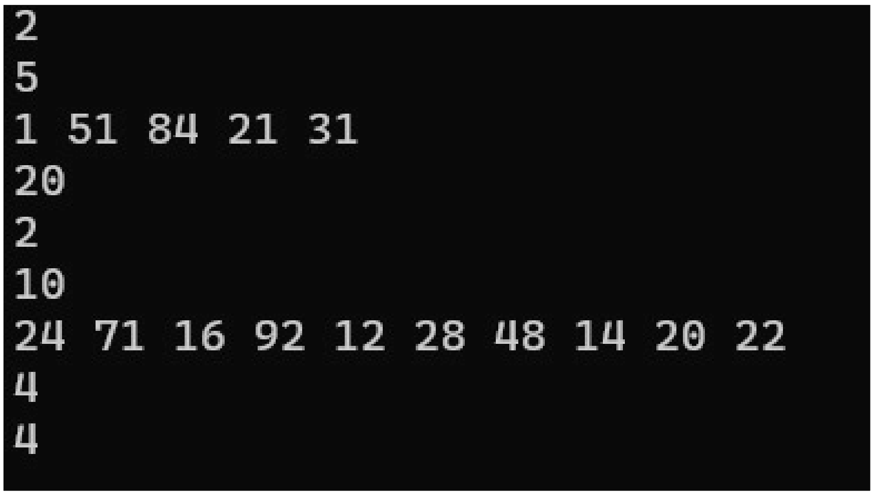


```
3
5
1 5 84 209 341
No sequence found.
10
24 28 48 71 86 89 92 120 194 201
2, 7, 8
15
64 69 82 95 99 107 113 141 171 350 369 400 511 590 666
1, 6, 9
```

III. 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.

```
#include<bits/stdc++.h>
using namespace std;
int main(){
int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;
                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                int k;  cin>>k;
```

```
                    int cnt=0;
                    for(int i=0;i<n;i++){
                            for(int j=0;j<n;j++){
                                    if(ar[i]-ar[j]==k) cnt++;
                            }
                    }
                    cout<<cnt<<"\n";
            }
return 0;
}
```



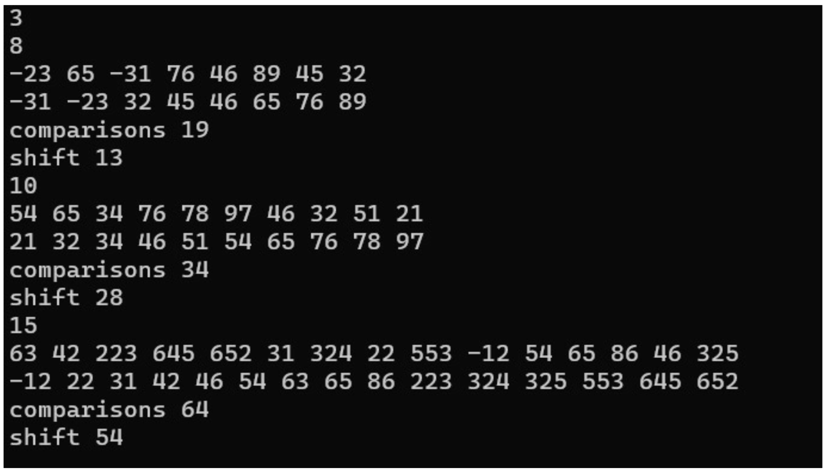
```
2
5
1 51 84 21 31
20
2
10
24 71 16 92 12 28 48 14 20 22
4
4
```

Week 3:

I. 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

```
#include<bits/stdc++.h>
using namespace std;
int main(){
  int t;
  cin>>t;
  while(t--){
    int n;
    cin>>n;
    vector<int> ar(n);
    for(int &i:ar) cin>>i;
    int compare=0;
    int shift=0;
    for(int i=1;i<n;i++){
      int temp=ar[i];
      int j=i-1;
      for(;j>=0;j--){
        compare++;
        if(ar[j]>temp){
          ar[j+1]=ar[j];
          shift++;
        }else break;
      }
      ar[j+1]=temp;
    }
    for(int i:ar) cout<<i<<" ";cout<<"\n";
```

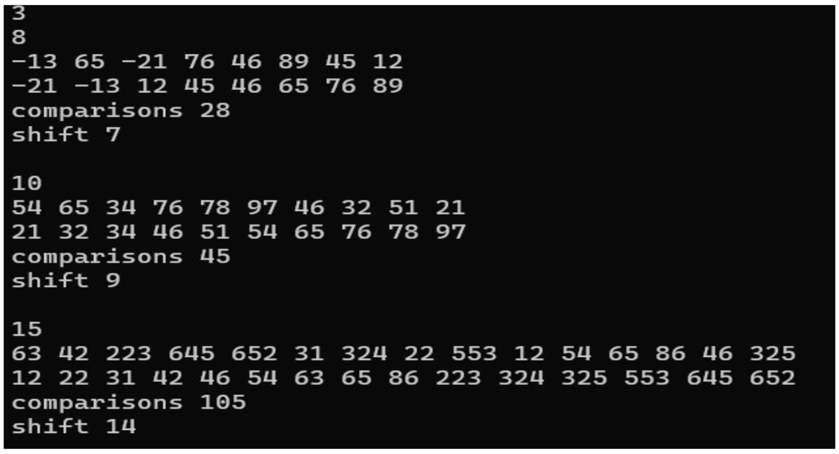
```
        cout<<"comparisons "<<compare<<"\n";
        cout<<"shift "<<shift<<"\n";
    }
    return 0;
}
```



```
3
8
-23 65 -31 76 46 89 45 32
-31 -23 32 45 46 65 76 89
comparisons 19
shift 13
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisons 34
shift 28
15
63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325
-12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
comparisons 64
shift 54
```

II. 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.

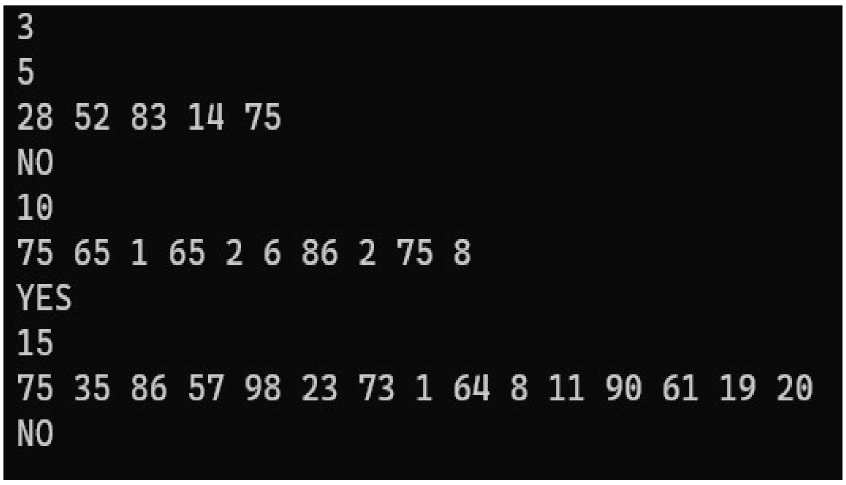
```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int t;
    cin>>t;
    while(t--){
        int n;
        cin>>n;
        vector<int> ar(n);
        for(int &i:ar) cin>>i;
        int compare=0;
        int shift=0;
        for(int i=0;i<n-1;i++){
            int mn=i;
            for(int j=i+1;j<n;j++){
                                    compare++;
                                    if(ar[mn]>ar[j]) mn=j;
                            }
                            shift++;
                            swap(ar[i],ar[mn]);
        }
        for(int i:ar) cout<<i<<" ";cout<<"\n";
        cout<<"comparisons "<<compare<<"\n";
        cout<<"shift "<<shift<<"\n";
    }
    return 0;
}
```



```
3
8
-13 65 -21 76 46 89 45 12
-21 -13 12 45 46 65 76 89
comparisons 28
shift 7

10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisons 45
shift 9

15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
comparisons 105
shift 14
```

III. 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))

```
#include<bits/stdc++.h>
using namespace std;
bool merge(vector<int>&arr,int  l,int mid,int r){
   bool found=0;
   int temp[r-l+2];
   int k=0;
   int i,j;
   i=l;
   j=mid+1;
   while(i<=mid  && j<=r){
      if(arr[i]==arr[j]) found=1;
      if(arr[i]>arr[j]){
         temp[k++]=arr[j++];
      }else temp[k++]=arr[i++];
   }
   while(i<=mid)temp[k++]=arr[i++];
   while(j<=r)temp[k++]=arr[j++];
   k=0;
   while(l<=r)
      arr[l++]=temp[k++];
      return  found;
}
bool split(vector<int>&arr,int  l,int r){
   if(l>=r)return  0;
   int mid = l + (r-l)/2;
   bool found = (split(arr,l,mid)  | split(arr,mid+1,r));
   return (found | merge(arr,l,mid,r));
}
int main(){
   int t;
   cin>>t;

   while(t--){
      int n;
      cin>>n;
      vector<int> ar(n);
      for(int &i:ar) cin>>i;

     if(split(ar,0,ar.size()-1))  cout<<"YES\n";
     else cout<<"NO\n";
   }
   return 0;
}
```

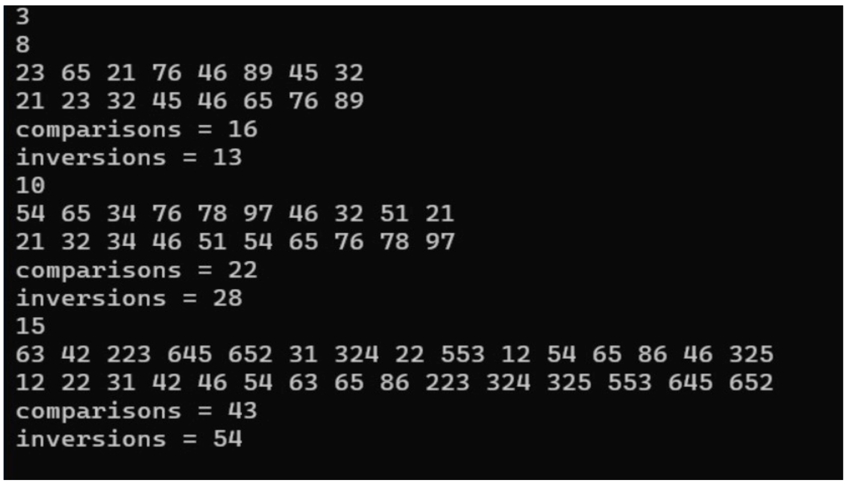


```
3
5
28 52 83 14 75
NO
10
75 65 1 65 2 6 86 2 75 8
YES
15
75 35 86 57 98 23 73 1 64 8 11 90 61 19 20
NO
```

Week 4:

I. 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.

```
#include<bits/stdc++.h>
using namespace std;
void merge(vector<int>&arr,int  l,int mid,int r,int&inv,int&comp){
   int temp[r-l+2];
   int k=0;
   int i,j;
   i=l;
   j=mid+1;
   while(i<=mid  && j<=r){
      comp++;
      if(arr[i]>arr[j]){
         inv+=mid-i+1;
         temp[k++]=arr[j++];
      }else temp[k++]=arr[i++];
   }
   while(i<=mid)temp[k++]=arr[i++];
   while(j<=r)temp[k++]=arr[j++];
   k=0;
   while(l<=r)   arr[l++]=temp[k++];
}
void split(vector<int>&arr,int  l,int r,int &inv,int &comp){
   if(l>=r)return;
   int mid = l + (r-l)/2;
   split(arr,l,mid,inv,comp);
   split(arr,mid+1,r,inv,comp);
   merge(arr,l,mid,r,inv,comp);
}
void count_inversions(vector<int>  &ar,int &inv,int &comp){
  split(ar,0,ar.size()-1,inv,comp);
}
int main(){

   int t;
   cin>>t;
  while(t--){
      int n;
      cin>>n;
      vector<int> ar(n);
      for(int &i:ar) cin>>i;
      int inv=0;
      int comp=0;
      count_inversions(ar,inv,comp);
      for(int i:ar) cout<<i<< " ";  cout<<"\n";
      cout<<"comparisons = "<<comp<<"\n";
      cout<<"inversions  = "<<inv<<"\n";
   }
   return 0;
}
```

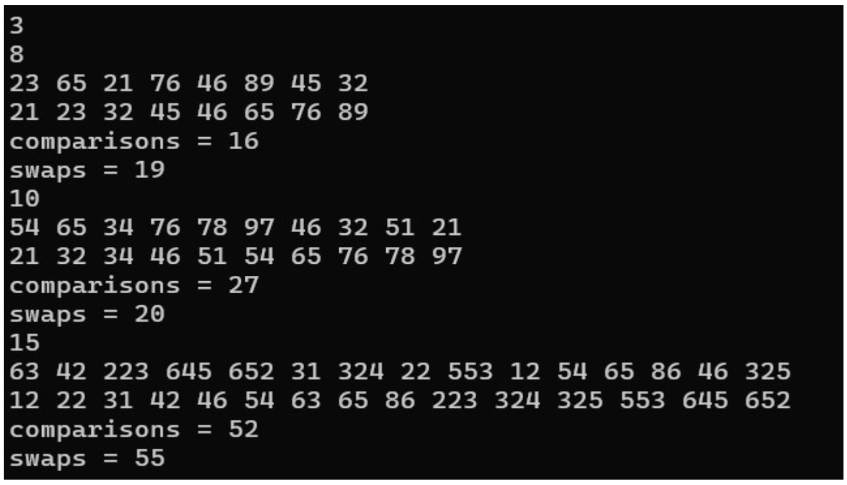


```
3
8
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
comparisons = 16
inversions = 13
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisons = 22
inversions = 28
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
comparisons = 43
inversions = 54
```

II. 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

```
#include<bits/stdc++.h>
using namespace std;
int partition(vector<int>  &ar,int &cmp,int &swp,int l,int r){
   int rx = l + rand()%(r-l);
   swp++;
   swap(ar[r],ar[rx]);
   int pivot = ar[r];
   int i = l-1;
   for(int j=l;j<=r-1;j++){
      cmp++;
      if (ar[j] < pivot)
      {
         i++;
         swp++;
         swap(ar[i],ar[j]);
      }
   }
   swp++;
   swap(ar[i+1],ar[r]);
   return i+1;
}
void quicksort(vector<int>  &ar,int &cmp,int  &swp,int l,int r){
   if(l>=r) return;
   int pi = partition(ar,cmp,swp,l,r);
   quicksort(ar,cmp,swp,l,pi-1);
   quicksort(ar,cmp,swp,pi+1,r);
}
int main(){
   int t;
   cin>>t;
   srand(time(0));
   while(t--){
      int n;
      cin>>n;
      vector<int> ar(n);
      for(int &i:ar) cin>>i;
```

```
        int cmp=0;
        int swp=0;
        quicksort(ar,cmp,swp,0,n-1);
        for(int &i:ar) cout<<i<<" ";
        cout<<"\n";
        cout<<"comparisons = "<<cmp<<"\n";
        cout<<"swaps = "<<swp<<"\n";
    }
    return 0;
}
```

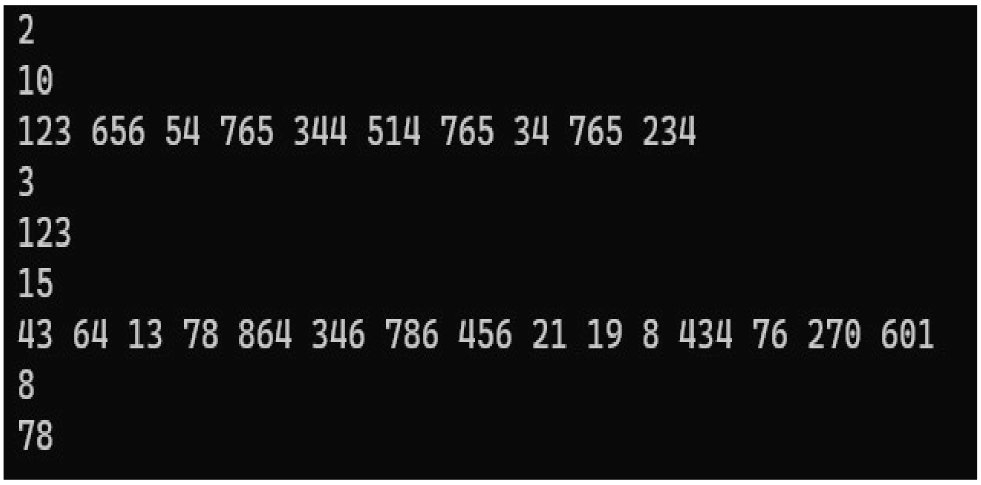


```
3
8
23 65 21 76 46 89 45 32
21 23 32 45 46 65 76 89
comparisons = 16
swaps = 19
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisons = 27
swaps = 20
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
comparisons = 52
swaps = 55
```

III. 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))

```
#include <bits/stdc++.h>
using namespace std;
int partition(int arr[], int l, int r){
        int x = arr[r], i = l;
        for (int j = l; j <= r - 1; j++){
                if (arr[j] <= x) {
                        swap(arr[i], arr[j]);
                        i++;
                }
        }
        swap(arr[i], arr[r]);
        return i;
}
int kthSmallest(int arr[], int l, int r, int k){
        if (k > 0 && k <= r - l + 1) {
                int index = partition(arr, l, r);
                if (index - l == k - 1)
                        return arr[index];
                if (index - l > k - 1)
                        return kthSmallest(arr, l, index - 1, k);
                return kthSmallest(arr, index + 1, r,k - index + l - 1);
        }

        return INT_MAX;
}
int main(){
        int t;
```

```
            cin>>t;
while(t--){
            int n,k;
    cin>>n;
    int arr[n];
    for(int i=0;i<n;i++) cin>>arr[i];
    cin>>k;
            cout<<kthSmallest(arr,  0, n - 1, k)<<"\n";
}
            return 0;
}
```

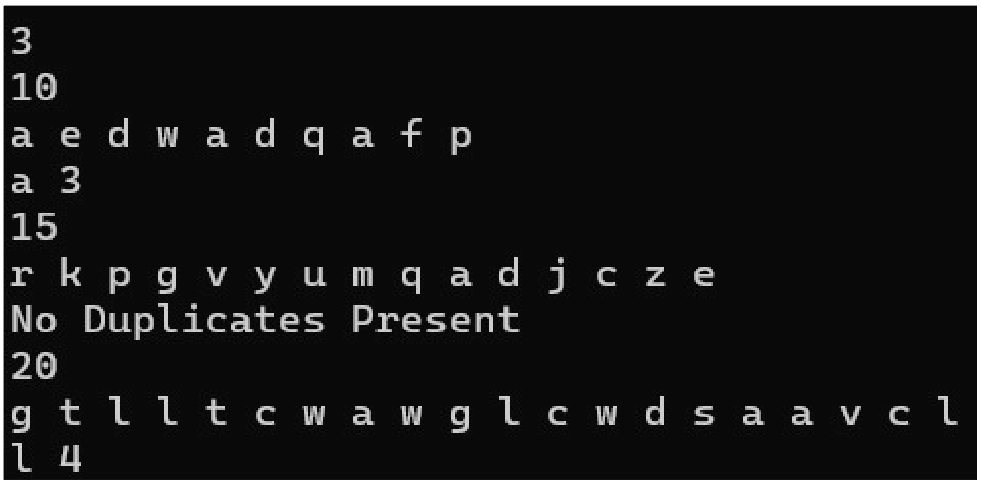


```
2
10
123 656 54 765 344 514 765 34 765 234
3
123
15
43 64 13 78 864 346 786 456 21 19 8 434 76 270 601
8
78
```

Week 5:

I. 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.

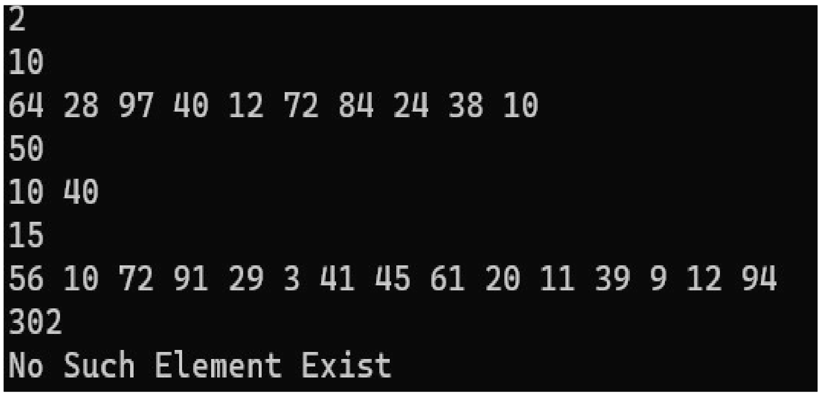
```
#include<bits/stdc++.h>
using namespace std;

int main(){
            int t;
            cin>>t;
            while(t--){
                        int n;
                        cin>>n;

                        vector<char> s(n);
                        for(char &c:s) cin>>c;

                        vector<int> cnt(26,0);
                        for(char &c:s) cnt[c-'a']++;
                        int mx=0;
                        for(int i=0;i<n;i++) if(cnt[mx]<cnt[i]) mx=i;

                        if(cnt[mx]==1) cout<<"No Duplicates Present\n";
                        else
                                    cout<<char('a'+mx)<<"   "<<cnt[mx]<<"\n";
            }
            return 0;
}
```



```
3
10
a e d w a d q a f p
a 3
15
r k p g v y u m q a d j c z e
No Duplicates Present
20
g t l l t c w a w g l c w d s a a v c l
l 4
```

II. 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))

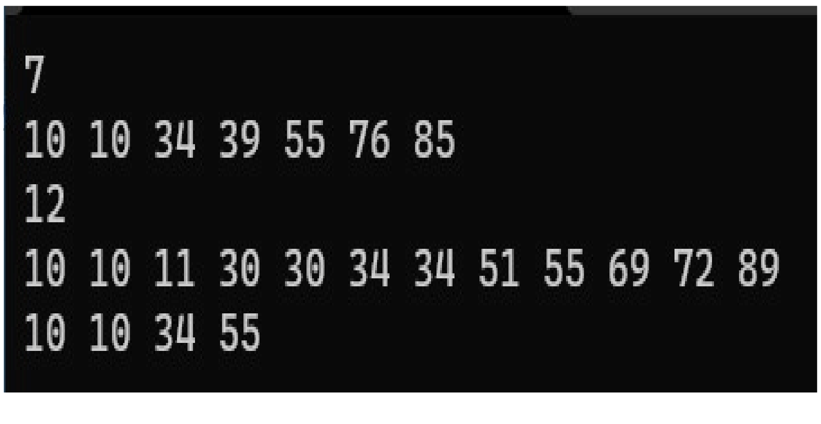
```
#include<bits/stdc++.h>
using namespace std;

void merge(vector<int>&arr,int  l,int mid,int r){
   int temp[r-l+2];
   int k=0;
   int i,j;
   i=l;
   j=mid+1;
   while(i<=mid && j<=r){
      if(arr[i]>arr[j]){
         temp[k++]=arr[j++];
      }else temp[k++]=arr[i++];
   }
   while(i<=mid)temp[k++]=arr[i++];
   while(j<=r)temp[k++]=arr[j++];
   k=0;
   while(l<=r)   arr[l++]=temp[k++];
}
void split(vector<int>&arr,int  l,int r){
   if(l>=r)return;
   int mid = l + (r-l)/2;
   split(arr,l,mid);
   split(arr,mid+1,r);
   merge(arr,l,mid,r);
}
int main(){
        int t;
        cin>>t;
        while(t--){
                int n;
                cin>>n;

                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                split(ar,0,n-1);

                int tar;
                cin>>tar;
                int i=0;
                int j=n-1;
                while(i<j){
                        int cur = ar[i] + ar[j];
                        if(cur==tar) break;
                        if(cur>tar) j--;
                        else i++;
                }
                if(i<j && ar[i]+ar[j]==tar) cout<<ar[i]<<" "<<ar[j]<<"\n";
                else cout<<"No Such Element Exist\n";

        }
        return 0;
}
```



```
2
10
64 28 97 40 12 72 84 24 38 10
50
10 40
15
56 10 72 91 29 3 41 45 61 20 11 39 9 12 94
302
No Such Element Exist
```

III. 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))

```
#include<bits/stdc++.h>
using namespace std;
int main(){
                int n;
                cin>>n;
                vector<int> ar(n);
                for(int &i:ar) cin>>i;
                int m;
                cin>>m;
                vector<int> ar2(m);
                for(int &i:ar2) cin>>i;
                int i=0;
                int j=0;
                while(i<n  && j<m){
                        if(ar[i]==ar2[j]) cout<<ar[i]<<" ",i++,j++;
                        else if(ar[i]>ar2[j]) j++;
                        else i++;
                }
        return 0;
}
```



```
7
10 10 34 39 55 76 85
12
10 10 11 30 30 34 34 51 55 69 72 89
10 10 34 55
```