Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

**Design And Analysis of Algorithms**

**Mid-Term Lab File**

**PCS – 409**

**B.TECH IV SEMESTER**

**WEEK - 1**

1. 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)

**SOURCE CODE**

```
#include<stdio.h>
int linearsearch(int arr[],int n,int key){
 int i,comp=0;
 for(i=0;i<n;i++){
  comp++;
  if(key==arr[i]){
     printf("Present %d",comp);
     return 1;
  }
 }
 printf("Not Present %d",comp);
 return 0;
}
int main(){
 int t,i,n,key;

 scanf("%d",&t);
 while(t--){
  scanf("%d",&n);
  int a[n];
  for(i=0;i<n;i++){
   scanf("%d",&a[i]);
  }
  scanf("%d",&key);
  linearsearch(a,n,key);
 }
 return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

2. 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(n log(n)), where n is the size of input).

**SOURCE CODE**

```
#include<iostream>
using namespace std;
int recursive_binary_search(int a[],int key,int l,int h){
  int mid;
  static int comp=0;
    if(l<=h){
      mid = (l+h)/2;
      comp++;
      if(a[mid]==key){
        cout<<"Present "<<comp<<endl;
        return 1;
      }
      if(a[mid]>key){
        return recursive_binary_search(a,key,l,mid-1);
      }
      else{
        return recursive_binary_search(a,key,mid+1,h);
      }
    }
    cout<<"Not Present "<<comp<<endl;
    return 0;
}
int main(){
  int t;
  cin>>t;
  while(t--){
    int n;
    cin>>n;
    int a[n];
    for(int i=0;i<n;i++){
      cin>>a[i];
    }
    int key;
    cin>>key;
    int l=0,h=n-1;
    recursive_binary_search(a,key,l,h);
  }
  return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

3. 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):

**SOURCE CODE**

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


void Jump_Search(int array[], int size, int key) {

   int comp=0;
   int start = 0;
   int m = sqrt(size);
   int i = m;

   while(array[i] <= key && i < size) {
      comp++;
      start = i;
      i += m;
      if(i > size - 1)
         i = size;
   }

   for(int j= start; j<i; j++) {
      comp++;
      if(array[j] == key){
         cout<<"Present :"<<comp;
         return;
      }
   }
   cout<<"Not Present: "<<comp;
   return;

}
```

```
int main(){
   int t;
   cin>>t;
   while(t--){
      int size;
      cin>>size;
      int arr[size];
      for(int i=0;i<size;i++){
         cin>>arr[i];
      }
      int key;
      cin>>key;
      Jump_Search(arr,size,key);
      cout<<endl;
   }
   return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

## WEEK - 2

1. 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”.

### SOURCE CODE

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

int countDuplicates(int a[], int key, int l, int h, int isFirst)
{
   int mid;
   int result = -1;
   while (l <= h)
   {
      mid = (l + h) / 2;
      if (a[mid] == key)
      {
         result = mid;
         if (isFirst == 1)
            h = mid - 1;
         else if (isFirst == 0)
            l = mid + 1;
      }
      else if (a[mid] > key)
         h = mid - 1;
      else
```

```
            l = mid + 1;
        }
        return result;
    }

    int main()
    {
        int t;
        cin >> t;
        while (t--)
        {
            int n;
            cin >> n;
            int a[n];
            for (int i = 0; i < n; i++)
                cin >> a[i];
            int key;
            cin >> key;
            int leftIndex = countDuplicates(a, key, 0, n - 1, 1);
            if (leftIndex == -1)
            {
                cout << "Not Found";
                return 0;
            }
            int rightIndex = countDuplicates(a, key, 0, n - 1, 0);
            int count = rightIndex - leftIndex + 1;
            cout << key << " - " << count;
        }

        return 0;
    }
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

2. 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”.

**SOURCE CODE**

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

int Search(int a[], int n, int key)
{
   int l = 0, h = n - 1;
   int mid;
   while (l <= h)
   {
      mid = (l + h) / 2;
      if (a[mid] == key)
         return mid;
      else if (a[mid] < key)
         l = mid + 1;
      else
         h = mid - 1;
   }
   return -1;
}
```

```
void findIndices(int a[], int n)
{
    for (int i = 0; i < n - 1; i++)
    {

      for (int j = i + 1; j < n; j++)
      {
        int sum = a[i] + a[j];
        int k = Search(a, n, sum);
       if (k != -1)
        {
           cout << i << "," << j << "," << k << endl;
           return;
        }
      }
   }
   cout << "No sequence found" << endl;
   return;
}
int main()
{
   int t;
   cin >> t;
   while (t--){
      int n;
      cin >> n;
      int a[n];
      for (int i = 0; i < n; i++)
         cin >> a[i];
      findIndices(a, n);
   }
   return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

3. 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.

**SOURCE CODE**

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

void sort(int a[], int n);

int main()
{
   int t;
   cin >> t;
   while (t--)
   {
      int n;
      cin >> n;
      int a[n];
      for (int i = 0; i < n; i++)
         cin >> a[i];
      int key;
      cin >> key;
      sort(a, n);
      int pair = 0;
      for (int i = 0; i < n - 1; i++)
      {
         for (int j = i + 1; j < n; j++)
         {
```

```
            if (a[j] - a[i] == key)
              pair++;
          }
        }
        cout << pair << endl;
    }
    return 0;
}


void sort(int a[], int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < (n - 1) - i; j++)
        {
            if (a[j] > a[j + 1])
            {
                swap(a[j], a[j + 1]);
            }
        }
    }
    return;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

## WEEK – 3

1. 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.

### SOURCE CODE

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

void InsertionSort(int[], int);

int main(){
   int t;
   cin>>t;
   while(t--)
   {
      int n;
      cin>>n;
      int a[n];
      for(int i=0;i<n;i++)
         cin>>a[i];
      InsertionSort(a,n);
   }
   return 0;
}
```

```
void InsertionSort(int a[], int n)
{
   int comp=0, shift=0;
   for(int i=1; i<n; i++)
   {
      int key = a[i];
      int j = i-1;
      comp++;
      while(a[j]>key && j>=0)
      {
         comp++;
         a[j+1] = a[j];
         shift++;
         j--;
      }
      a[j+1] = key;
   }
   for(int i=0;i<n;i++)
      cout<<a[i]<<" ";
   cout<<endl;
   cout<<"Comparisons = "<<comp<<endl;
   cout<<"Shift = "<<shift<<endl;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

2. 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.

**SOURCE CODE**

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

void SelectionSort(int [],int);

int main()
{
   int t;
   cin>>t;
   while(t--)
   {
      int n;
      cin>>n;
      int a[n];
      for(int i=0;i<n;i++)
      {
         cin>>a[i];
      }
      SelectionSort(a,n);
   }
   return 0;
}
```

```
void SelectionSort(int a[], int n)
{
   int comp=0, swaps=0;
   for(int i=0;i<n-1;i++)
   {
      int min = i;
      for(int j=i+1;j<n;j++)
      {
         comp++;
         if(a[j]<a[min])
            min = j;
      }
      swap(a[i],a[min]);
      swaps++;
   }
   for(int i=0;i<n;i++)
      cout<<a[i]<<" ";
   cout<<endl;
   cout<<"Comparisons = "<<comp<<endl;
   cout<<"Swaps = "<<swaps<<endl;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

3. 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'.

**SOURCE CODE**

```
#include<iostream>
using namespace std;
void mergeSort (int a[], int l, int r);
void merge(int a[], int l, int mid, int r);
void findDuplicate(int a[], int n)  {
   bool flag = false;
   for(int i=0;i<n-1;i++)  {
      int prev = a[i];
      for(int j=i+1;j<n;j++){
         if(a[j]==prev){
            flag = true;
            break;
         }
         prev = a[j];
      }
      if(flag)  {
         cout<<"YES"<<endl;  break;
      }
   }
   if(!flag)
      cout<<"NO"<<endl;
}
int main(){
   int t;
   cin>>t;
   while(t--){
      int n;
```

```
        cin>>n;
        int a[n];
        for(int i=0;i<n;i++)
            cin>>a[i];
        mergeSort(a,0,n-1);
        findDuplicate(a,n);
    }
    return 0;
}
void mergeSort(int a[], int l, int r){
    if(l<r){
        int mid = (l+r)/2;
        mergeSort(a,l,mid);
        mergeSort(a,mid+1,r);
        merge(a,l,mid,r);
    }
}
void merge(int a[], int l, int mid, int r){
    int c[r];
    int i = l, j = mid+1, k = l;
    while(i<=mid && j<=r){
        if(a[i]<a[j]){
            c[k] = a[i];
        }
        else{
            c[k] = a[j];
        }
    }
    while(i<=mid){
        c[k] = a[i];
    }
    while(j<=r){
        c[k] = a[j];
    }
    for(k=l;k<=r;k++)
        a[k]=c[k];
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

## WEEK – 4

1. 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

### SOURCE CODE

```
#include<iostream>
using namespace std;
int comp=0;
int merge(int a[], int l, int mid, int r){
   int countInversions = 0;
   int c[r];
   int i = l,  j = mid;
   int k = l;
   while(i<mid && j<=r){
      if(a[i]<a[j]){
         c[k++] = a[i++];
      }
      else{
         c[k++] = a[j++];
         countInversions = countInversions + (mid-i);
      }
      comp++;
   }
   while(i<mid){
      c[k++] = a[i++];
   }
   while(j<=r){
      c[k++] = a[j++];
   }
   for(k=l;k<=r;k++)
      a[k]=c[k];
   return countInversions;
}
```

```
int mergeSort(int a[], int l, int r){
   int inversions=0;
   if(l<r){
      int mid = (l+r)/2;
      inversions = mergeSort(a,l,mid);
      inversions += mergeSort(a,mid+1,r);
      inversions += merge(a,l,mid+1,r);
   }
   return inversions;
}

int main(){
   int t;
   cin>>t;
   while(t--)
   {
      int n;
      cin>>n;
      int a[n];
      for(int i=0;i<n;i++)
         cin>>a[i];
      int inversions = mergeSort(a,0,n-1);
      for(int i=0;i<n;i++)
         cout<<a[i]<<" ";
      cout<<endl;
      cout<<"Comparisons = "<<comp<<endl;
      cout<<"Inversions = "<<inversions<<endl;
   }
   return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

2. 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.

**SOURCE CODE**

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

int partition(int a[],int low,int high, int & comp, int & swaps){
   int pivot = a[low];
   int i = low+1;
   int j = high;
   do{
      while(a[i]<=pivot){
         comp++;
         i++;
      }
      while(a[j]>pivot){
         comp++;
         j--;
      }
      if(i<j){
         swap(a[i],a[j]);
         swaps++;
      }
   }while(i<j);
   swap(a[low],a[j]);
   swaps++;
   return j;
}
```

```
void quickSort(int a[],int low,int high,int & comp, int & swaps){
   if(low<high){
      int partitionIndex = partition(a,low,high,comp,swaps);
      quickSort(a,low,partitionIndex-1,comp,swaps);
      quickSort(a,partitionIndex+1,high,comp,swaps);
   }
}

int main()
{
   int t;
   cin>>t;
   while(t--){
      int n;
      cin>>n;
      int a[n];
      for(int i=0;i<n;i++)
         cin>>a[i];
      int comp = 0,swaps = 0;
      quickSort(a,0,n-1,comp,swaps);
      for(int i=0;i<n;i++)
         cout<<a[i]<<" ";
      cout<<endl;
      cout<<"Comparisons = "<<comp<<endl;
      cout<<"Swaps = "<<swaps<<endl;
   }
   return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

3. 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))

**SOURCE CODE**

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

int partition(int a[],int low,int high){
   int pivot = a[low];
   int i = low+1;
   int j = high;
   do{
      while(a[i]<=pivot)
         i++;
      while(a[j]>pivot)
         j--;
      if(i<j)
         swap(a[i],a[j]);
   }while(i<j);
   swap(a[low],a[j]);
   return j;
}
void quickSortSmallest(int a[],int low,int high,int n,int k){
   if(low<high){
      int partitionIndex = partition(a,low,high);
      if(partitionIndex == k)
      {
         cout<<k<<"th Smallest: "<<a[k-1]<<endl;
         cout<<k<<"th Largest: "<<a[n-k+1]<<endl;
      }
      else if(k<partitionIndex)
         quickSortSmallest(a,low,partitionIndex-1,n,k);
      else if(k>partitionIndex)
         quickSortSmallest(a,partitionIndex+1,high,n,k);
   }
}
```

```
int main(){
   int t;
   cin>>t;
   while(t--)
   {
      int n;
      cin>>n;
      int a[n];
      for(int i=0;i<n;i++)
         cin>>a[i];
      int k;
      cin>>k;
      quickSortSmallest(a,0,n-1,n,k);

   }
   return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

## WEEK - 5

1. 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)

## SOURCE CODE

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

int main()
{
   int t;
   cin>>t;
   while(t--)
   {
      int n;
      cin>>n;
      char a[n];
      for(int i=0;i<n;i++)
         cin>>a[i];
      char max = a[0];
      for(int i=1;i<n;i++)
      {
         if(a[i]>max)
            max = a[i];
      }

      int count[max+1];
      for(int i=65;i<max+1;i++)
      {
         count[i] = 0;
      }
```

```
        for(int i=0;i<n;i++)
        {
           count[a[i]] = count[a[i]] + 1 ;
        }

        int maxOccur = 65;

        for(int i=66;i<max+1;i++)
        {
           if(count[i]>count[maxOccur])
              maxOccur = i;
        }
        if(count[maxOccur]>1)
           cout<<(char)maxOccur<<" - "<<count[maxOccur]<<endl;
        else
           cout<<"No Duplicates Found"<<endl;

         return 0;

    }
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

2. 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))

**SOURCE CODE**

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

void merge(int a[], int low, int mid, int high){
   int c[high];
   int i = low;
   int j = mid+1;
   int k = low;
   while(i<=mid && j<=high){
      if(a[i]<a[j]){
         c[k] = a[i];
         k++;i++;
      }
      else{
         c[k] = a[j];
         k++;j++;
      }
   }
   while(i<=mid){
      c[k] = a[i];
      k++;i++;
   }
   while(j<=high){
      c[k] = a[j];
      k++;j++;
   }
   for(k=low;k<=high;k++)
      a[k]=c[k];
}
```

```
void mergeSort(int a[], int low, int high){
   if(low<high){
      int mid = (low+high)/2;
      mergeSort(a,low,mid);
      mergeSort(a,mid+1,high);
      merge(a,low,mid,high);
   }
}

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

      int i=0,j=n-1;
      while(i<j)
      {
         if(a[i]+a[j]==key){
            cout<<a[i]<<" "<<a[j]<<endl;
            break;
         }
         else if(a[i]+a[j]>key)
            j--;
         else
            i++;
      }
      if(i>=j)
         cout<<"No Such Element Exists"<<endl;
   }
   return 0;
}
```

Name – Anshul Kumar
Sec – CST SPL-2
Roll no - 11

3. 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))

**SOURCE CODE**

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

void CountSort(int a[], int n)
{
   int max = a[0];
      for(int i=1;i<n;i++)
      {
         if(a[i]>max)
            max = a[i];
      }
      int count[max+1];
      for(int i=0;i<max+1;i++)
      {
         count[i] = 0;
      }
      for(int i=0;i<n;i++)
      {
         count[a[i]] = count[a[i]] + 1 ;
      }
      int i=0,j=0;
      while(i<max+1)
      {
         if(count[i]>0)
         {
            a[j] = i;
            count[i] = count[i] - 1 ;
            j++;
         }
         else{
            i++;
         }
      }
```

```
    for(int i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;

}

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

   CountSort(a,m);
   CountSort(b,n);

   int i=0,j=0;

   while( i<m && j<n )
   {
      if(a[i]==b[j]){
         cout<<a[i]<<" ";
         i++; j++;
      }
      else if(a[i]<b[j])
         i++;
      else
         j++;
   }

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