Practical 1

Problem statement: 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).

Algorithm: To find whether given key element is present in the array or not.

Step 1: Start

Step 2: linearsearch(arr,key)

1-Initialize count to 0

2-Loop through each element in arr from index 0 to n-1.

Increment count.

If arr[i] equals key:

a- Print "Present ".

b-Exit the loop.

3-If the key was not found print “not present”.

Step 3: main()

1-Read the test cases.

2-For each test case:

a- Read the number of elements.
b- Declare an array arr of size n.
c- Read n elements of arr.
d- Read the search key.

3-Call the function linearsearch() to find whether given key element is present in the array or not.

Step 4: Stop

Source Code:

```
#include <iostream>
#include <vector>
using namespace std;
void linearSearch(const vector<int>& arr, int key) {
    int count = 0;
    for (size_t i = 0; i < arr.size(); i++) {
        count++;
        if (arr[i] == key) {
            cout << "Present " << count<< endl;
            return;
        }}
    cout << "Not Present " << count << endl;
}int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        vector<int> arr(n);
        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }
        int key;
        cin >> key;
        linearSearch(arr, key);
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\"
 { .\week11 }
3
8
34 35 65 31 25 89 64 30
89
Present 6
5
977 354 244 546 355
244
Present 3
6
23 64 13 67 43 56
63
Not Present 6
```

Practical 2

Problem statement : 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)

Algorithm: To find whether given key element is present in the array or not.

Step 1: Start

Step 2:binarysearch(arr,key,count)

1- initialize start = 0 and end = n - 1.

2- Repeat the following while start <= end

a- Calculate mid = start + (end - start) / 2.

b- Increment count

c-If arr[mid] == key, return mid

d-If arr[mid] < key, update start = mid + 1

e-Else, update end = mid - 1

3- If the key is not found, return -1.

Step 3: main()

1-Read the test cases.

2-For each test case:

a- Read the number of elements.

b- Declare an array arr of size n.

c- Read n elements of arr.

d- Read the search key.

3-Call the function binarysearch() to find whether given key element is present in the array or not.

Step 4: Stop

Source Code:

```
#include <iostream>
#include <vector>
using namespace std;
int binarySearch(vector<int>& arr, int key, int& count) {
    int start = 0, end = arr.size() - 1;
    while (start<= end) {
        int mid = start + (end - start) / 2;
        count++;
        if (arr[mid] == key) {
            return mid;
        } else if (arr[mid] < key) {
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return -1;
}
int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        vector<int> arr(n);
        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }
```

```
        int key;
        cin >> key;
        int count = 0;
        int result = binarySearch(arr, key, count);

        if (result != -1)
            cout << "Present " << count << endl;
        else
            cout << "Not Present " << count << endl;
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\"
 { .\week12 }
3
8
34 35 65 31 25 89 64 30
89
Present 2
5
977 354 244 546 355
244
Present 1
6
23 64 13 67 43 56
63
Not Present 3
```

Practical 3

Problem statement: 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)

Algorithm: To find whether a given key element is present in the sorted array or not.

Step1: Start

Step2: jumpsearch(arr,n,key.count)

1- Initialize jump step m=sqrt(n)
2- Initialize i=0
3- Set count=0
4- While arr[min(m, n) - 1] < key
   a- Increment count.
   b-Move to the next block: i = m and m += sqrt(n).
   c- If i >= n, return -1
5- Perform linear search in the identified block:
   a- While i < min(m, n) and arr[i] < key
      1- Increment count.
      2-Move to the next index i++

6- Increment count (final comparison).
7- If arr[i] == key, return i (key found).
8- Otherwise, return -1

Step3: main()

1-Read the test cases.

2-For each test case:

a- Read the number of elements.

b- Declare an array arr of size n.

c- Read n elements of arr.

d- Read the search key.

3-Call the function jumpsearch() to find whether given key element is present in the

Array or not

Step 4: Stop

Source code:

```
#include <iostream>
#include <cmath>
using namespace std;
int jumpSearch(int arr[], int n, int key,int&count) {
    int m= sqrt(n);
    int i = 0;
    count = 0;
    while (arr[min(m, n) - 1] < key) {
        count++;
        i= m;
        m+= sqrt(n);
        if (i>= n) {
            return -1;
        }}
    while (i< min(m, n) && arr[i] < key) {
        count++;
        i++;
    }
    count++;
    if (arr[i] == key) {
        return i;
    }
    return -1; }
int main() {
    int t;
    scanf("%d",&t);
    while(t--){
    int n;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++) {
        cin >> arr[i];}
    int key, count = 0;
    cin >> key;
    int result = jumpSearch(arr, n, key, count);
    if (result != -1) {
        cout << "Present "<<count<<endl;
    } else {
        cout << "Not Present " <<count<< endl;}
}
    return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\" ;
eek13 } ; if ($?) { .\week13 }
3
5
12 23 36 39 41
41
Present 3
8
21 39 40 45 51 54 68 72
69
Not Present 5
10
101 246 438 561 796 896 899 4644 7999 8545
7999
Present 5
```

Practical 4

Problem statement: 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)).

Algorithm: To find whether the given key element is present in the array or not.

Step1: Start

Step 2: firstoccurrence(arr, key)

1-Initialize low = 0, high = n - 1, and result = -1.

2-While low ≤ high:

a-Compute mid = low + (high - low) / 2.

b-If arr[mid] == key:

1-Update result = mid (store the first occurrence).

2-Search for an earlier occurrence: high = mid - 1.

c-If arr[mid] < key, move to the right: low = mid + 1.

d-Otherwise, move to the left: high = mid - 1.

3-Return result.

Step 3: lastoccurrence(arr,key)

1-Initialize low = 0, high = n - 1, and result = -1.

2-While low ≤ high:

a-Compute mid = low + (high - low) / 2.

b-If arr[mid] == key:

1-Update result = mid (store the first occurrence).

2-Search for an earlier occurrence: high = mid - 1.

c-If arr[mid] < key, move to the right: low = mid + 1.

d-Otherwise, move to the left: high = mid - 1.

3-Return result.

Step 4: Repeat the following steps for each test case:

1-Read the number of elements.

2-Declare a vector of size n.

3-Read n.

4-Read the key element to be searched.

5-Call function firstoccurrence(),lastoccurrence().

6- If first == -1, print "Key not present".

7- Otherwise, occurrence = last - first + 1 and print key - occurrence.

Step 5: Stop

Source code:

```
#include <iostream>
#include <vector>
using namespace std;
int firstoccurrence(const vector<int>& arr, int key) {
    int low = 0, high = arr.size() - 1, result = 0;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == key) {
            result = mid;
            high = mid - 1;
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return result;
}
int lastoccurrence(const vector<int>& arr, int key) {
    int low = 0, high = arr.size() - 1, result = 0;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == key) {
            result = mid;
            low = mid + 1;
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
```

```
            high = mid - 1;
        }
    }
    return result;
int main() {
    int T;
    cin >> T;
    while (T--) {
        int n, key;
        cin >> n;
        vector<int> arr(n);
        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }
        cin >> key;
        int first = firstoccurrence(arr, key);
        int last = lastoccurrence(arr, key);
         int occurence=last-first+1;
        if (first == -1) {
            cout << "Key not present" << endl;
        } else {
            cout << key << " - " << occurence << endl;
        }
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\"
eek21 } ; if ($?) { .\week21 }
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
```

Practical 5

Problem statement: 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].

Algorithm: To find three indices i, j, k such that arr[i] + arr[j] = arr[k].

Step 1: findindices(arr, n)

1-Iterate k from 3 to n:

a-Initialize i = 1 (first index) and j = k - 1 (last index before k).

b-While i < j:

1-Compute sum = arr[i] + arr[j].

2-If sum == arr[k]:

Print indices i, j, k and return.

3-If sum < arr[k]:

Move i right (i++).

4-Otherwise:

Move j left (j--).

c-print no sequence found

Step 2: Repeat the following steps for each test case:

1-Read the number of elements in the array (n).

2- Declare a vector of size n.

3-Read n elements

4-Call function findindices().

Step 3: Stop

Source code:

```
#include <iostream>
#include <vector>
using namespace std;
void findindices(vector<int>& arr, int n) {
    for (int k = 3; k <=n; k++) {
        int i = 1, j = k - 1;
        while (i < j) {
            int sum = arr[i] + arr[j];
            if (sum == arr[k]) {
                cout << i << ", " << j << ", " << k << endl;
                return;
            } else if (sum < arr[k]) {
                i++;
            } else {
                j--;
            }  } }
    cout << "No sequence found." << endl;}
int main() {
    int T;
    cin >> T;
    while (T--) {
        int n;
        cin >> n;
        vector<int> arr(n);
        for (int i = 1; i <= n; i++) {
            cin >> arr[i];}
        findindices(arr, n);}
    return 0;}
```

OUTPUT:


PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\"
} ; if ($?) { .\q22 }
2
5
1 5 85 209 341
No sequence found.
10
24 28 48 71 86 89 92 120 194 201
2, 7, 8

Practical 6

Problem statement: 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.

Algorithm: To count the number of pairs of integers such that their difference is equal to a given key.

Step 1: Start

Step 2: Read the number of test cases (T).

Step 3: Repeat the following steps for each test case:

1-Read the number of elements in the array (n).

2-Declare an array of size n.

3-Read n elements of the array.

4-Read the key.

Step 4: Initialize count = 0.

Step 5: Iterate i from 0 to n - 2:

1-Iterate j from i + 1 to n - 1:

a-If arr[j] - arr[i] == key or arr[i] - arr[j] == key, increment count.

Step 6: Print count (the number of pairs satisfying the condition).

Step 7:Stop

Source code:

```
#include <iostream>
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;
      int count = 0;
      for (int i = 0; i < n - 1; i++)
      {
         for (int j = i + 1; j < n; j++)
          {
            if (arr[j] - arr[i] == key || arr[i] - arr[j] == key) {
               count++;
            }} }
      cout<< count<<endl;
   }
   return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\"
weekq23 } ; if ($?) { .\weekq23 }
2
5
1 51 84 21 31
20
2
10
24 71 16 92 12 28 48 14 20 22
4
4
```

Practical 7

Problem statement: 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.

Algorithm: To find number of comparisons and shifts.

Step 1: Start

Step 2: insertion(v)

loop from j = 1 to n-1:

1. Store temp = v[j].
2. Increment shift (storing temp is a shift).
3. Initialize i = j - 1.
4. while i >= 0 and v[i] > temp:

a-Shift v[i] to v[i+1].

b-Increment count (comparison).

c-Increment shift (shift operation).

d-Decrement i.

e-Place temp at v[i+1].

5- print the sorted array.

6-print the total number of comparisons and shifts.

Step 2: read the number of test cases t.

1- For each test case:

a-read the number of elements n.

b-Initialize a vector v of size n and input n elements.

2-Initialize:

a-count = 0

b-shift = 0

Source Code:

```
#include<iostream>
#include<vector>
using namespace std;
void insertion(vector<int>& v) {
    int count = 0;
    int shift = 0;
    for (int j = 1; j < v.size(); j++) {
        int i = j - 1;
        int temp = v[j];
        shift++;
        while (i >= 0 && v[i] > temp) {
            v[i + 1] = v[i];
            count++;
            i = i - 1;
            shift++;
        }
        v[i + 1] = temp;
    }
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
    cout << "comparisons = " << count << endl;
    cout << "shifts = " << shift << endl;
}
int main() {
    int t, n;
    cin >> t;
```

```
    for (int i = 0; i < t; i++) {
        cin >> n;
        vector<int> v(n);
        for (int j = 0; j < n; j++) {
            cin >> v[j];
        }
        insertion(v);
    }
    return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\" ;
 { .\week31 }
3
8
-23 65 -31 76 46 89 45 32
-31 -23 32 45 46 65 76 89
comparisons = 13
shifts = 20
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisons = 28
shifts = 37
15
63 42 53 645 652 31 324 22 553 -12 54 65 86 46 325
-12 22 31 42 46 53 54 63 65 86 324 325 553 645 652
comparisons = 52
shifts = 66
```

Practical 8

Problem statement: 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.

Algorithm: To sort an array using Selection Sort

Step 1: Start

Step 2: selectionSort(arr, n, comparisons, swaps)

1. For each element i from 0 to n-2, set minIndex = i.
2. For each element j from i+1 to n-1,
   a.increment comparisons by 1.
   b-if arr[j] < arr[minIndex], set minIndex = j.
3. after the inner loop:
   a.if minIndex != i, swap arr[i] and arr[minIndex] and increased swap by 1.

Step 3 : main()

1. read the number of test cases T.
2. For each test case:
   a. read the number of elements and store elements in array.
   b. call selectionSort(arr, n, comparisons, swaps) to sort the array.
3. print the sorted array.
4. Print the number of comparisons and swaps.

Step 4: Stop

Source code:

```
#include <iostream>
using namespace std;
void selectionSort(int arr[], int n, int &count, int &swaps) {
    count = 0;
    swaps = 0;
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            count++;
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        if (minIndex != i) {
            swap(arr[i], arr[minIndex]);
            swaps++;
        }
    }
}

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 count, swaps;
    selectionSort(arr, n, count, swaps);
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << "\ncomparisons = " << count << "\nswaps = " << swaps << endl;
  }
  return 0;
}
```

OUTPUT:

```
PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35\"
eek32 } ; if ($?) { .\week32 }
2
8
-13 65 -21 76 46 89 45 12
-21 -13 12 45 46 65 76 89
comparisons = 28
swaps = 5
10
54 65 34 76 78 97 46 32 51 21
21 32 34 46 51 54 65 76 78 97
comparisons = 45
swaps = 6
```

Practical 9

Problem statement: 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).

Algorithm: To find duplicates in array using quick sort.

Step 1: Start

Step 2: partition(arr, low, high)

1. set the pivot element as the last element of the array and variable i to low-1.
2. iterate over the array from index low to high - 1:
    a. if the current element (arr[j]) is smaller than the pivot,then increment i and swap arr[i] and arr[j].
3. after the loop ends, swap the pivot (arr[high]) with arr[i + 1] to place the pivot in its correct position.
4. return i + 1 as the partition index.

Step 3: quickSort(arr, low, high)

1. if low < high:
    a.call partition(arr, low, high) to find the partition index (p).
    b..recursively call quickSort on the left subarray and right subarray

Step 4: duplicates(arr, n)

1. call quickSort(arr, 0, n - 1) to sort the array.
2. iterate over the array from index 1 to n - 1:
    a.if arr[i] == arr[i - 1], return true, if duplicate found.
3. if no duplicates are found, return false.

Step 5: Main()

1. read the number of test cases.
2. For each test case:
    a.read the number of elements in the array and array elements.
    b.call duplicates(arr, n)
3. if duplicates are found, print "Yes" and if no found, print “no”.

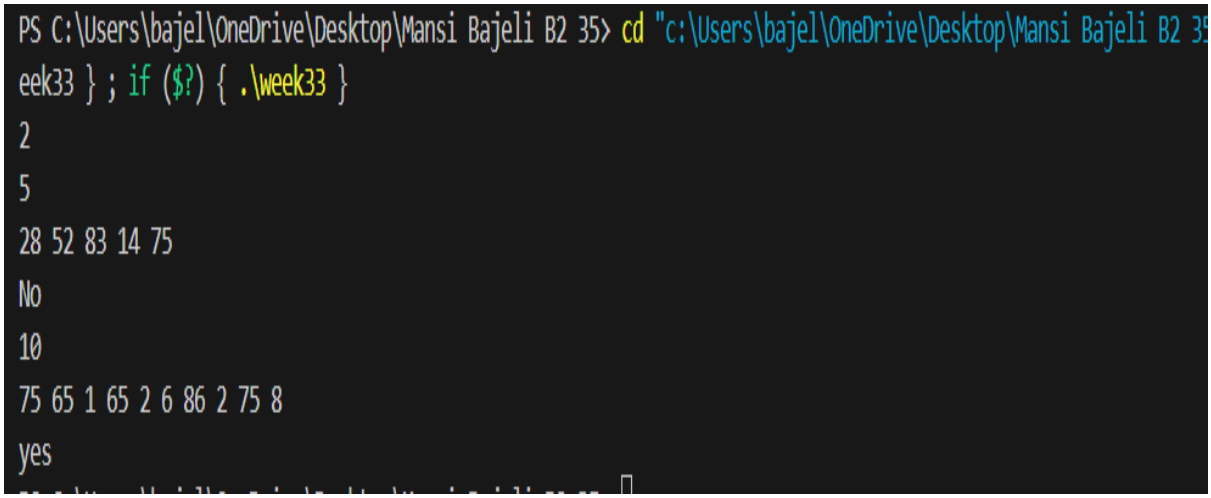
Step 6: Stop

Source code :

```
#include <iostream>
#include <vector>
using namespace std;
int partition(vector<int>& arr, int low, int high) {
    int pivot=arr[high];
    int i=low-1;
    for(int j=low;j<high;j++){
        if(arr[j]<pivot){
            i++;
            swap(arr[i],arr[j]);
        }
    }
    swap(arr[i+1],arr[high]);
    return(i+1);
}
void quickSort(vector<int>& arr, int low, int high) {
    if(low<high){
        int p=partition(arr,low,high);
        quickSort(arr, low, p-1);
        quickSort(arr, p+1, high);
    }
}
bool duplicates(vector<int>& arr,int n){
    quickSort(arr,0,n-1);
    for(int i=1;i<n;i++){
        if(arr[i] == arr[i-1]){
            return true;
        }
```

```
    }
    return false;
}
int main() {
    int testcases;
    cin>>testcases;
    for(int t=0;t<testcases;t++){
        vector<int> arr;
        int n,num;
        cin>>n;
        for(int i=0;i<n;i++){
            cin>>num;
            arr.push_back(num);
        }
        if(duplicates(arr,n)){
            cout<<"yes"<<endl;
        } else {
            cout<<"No"<<endl;
        }
    }
    return 0;
}
```

OUTPUT:


PS C:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 35> cd "c:\Users\bajel\OneDrive\Desktop\Mansi Bajeli B2 3
eek33 } ; if ($?) { .\week33 }
2
5
28 52 83 14 75
No
10
75 65 1 65 2 6 86 2 75 8
yes