	Marwari University **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing the Searching Algorithms	
Experiment No: 02	**Date:**	**Enrollment No: 92200133030**

Aim: Implementing the Searching Algorithms

IDE: Visual Studio Code

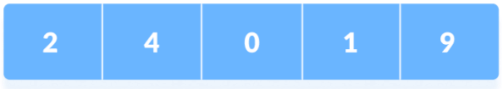
Linear Search :-

Theory: -

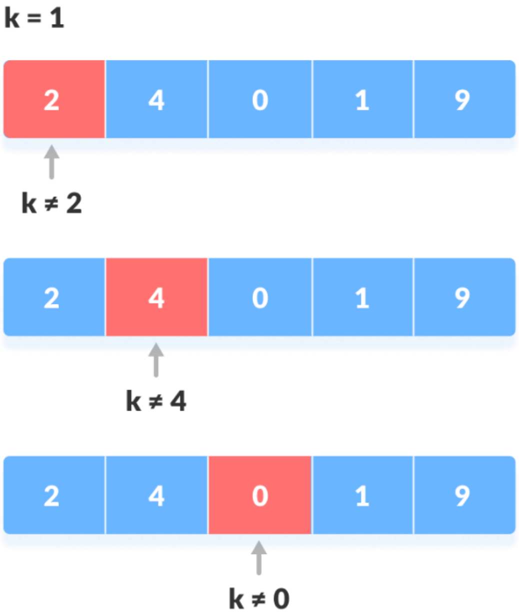
- Linear search is a sequential searching algorithm where we start from one end and check every element of the list until the desired element is found. It is the simplest searching algorithm.

Working of Linear Search :-

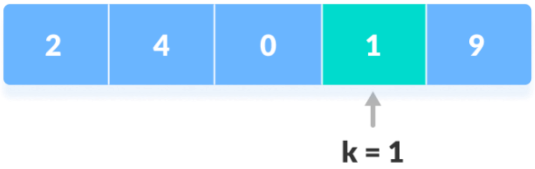
- The following steps are followed to search for an element k = 1 in the list below.



1) Start from the first element, compare k with each element x.



2) If x == k, return the index.



3) Else, return not found.

Algorithm: -

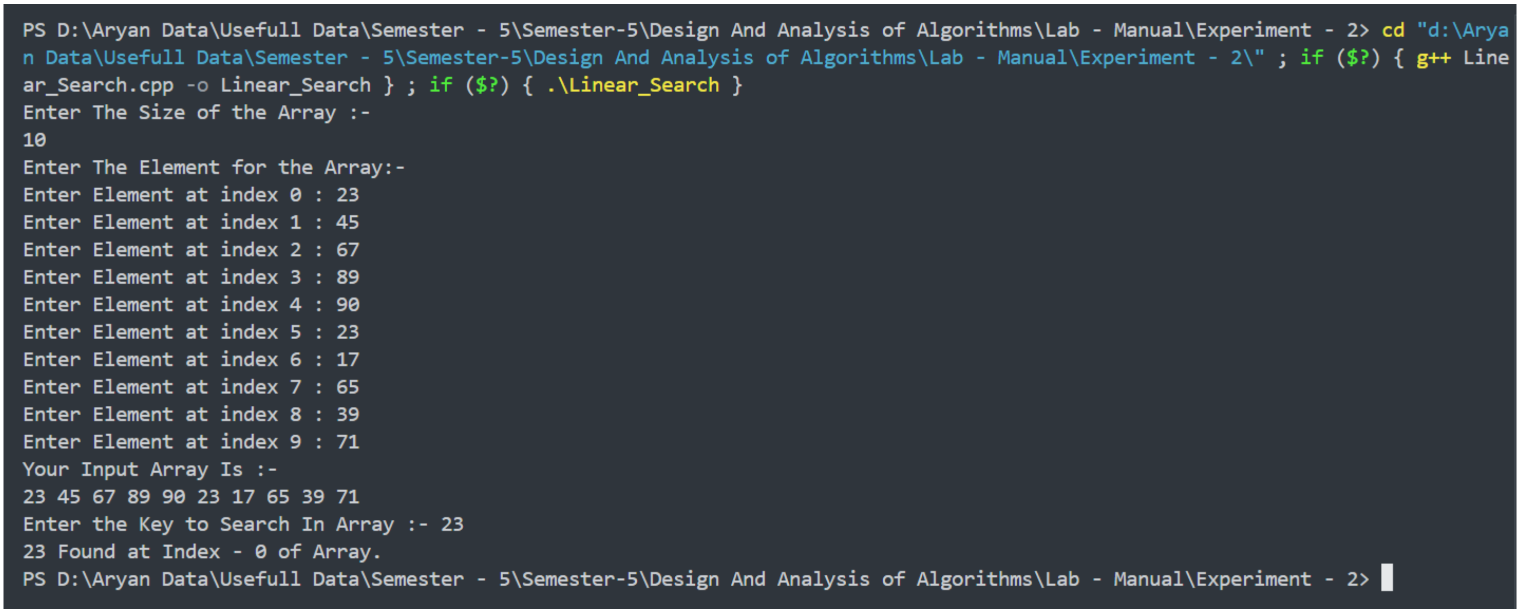
Programming Language: - C++

Code :-

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

int Linear_Search(vector<int> Array, int key) {
    for (int i = 0; i < Array.size(); i++) {
        if (Array[i] == key) {
            return i;
        }
    }
    return -1;
}
```

```
void Print_Array(vector<int> Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << Array[i] << " ";
    }
    cout << endl;
}

void Input_Array(vector<int>& Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << "Enter Element at index " << i << " : ";
        cin >> Array[i];
    }
}

int main() {
    int size;
    int key;

    while (true) {
        cout << "Enter The Size of the Array :- " << endl;
        cin >> size;

        if (size >= 1) {
            break;
        }

        cout << "Invalid Size. Size must be a Positive Integer." << endl;
    }

    vector<int> Array(size, 0);
    cout << "Enter The Element for the Array:- " << endl;
    Input_Array(Array);
    cout << "Your Input Array Is :- " << endl;
    Print_Array(Array);
    cout << "Enter the Key to Search In Array :- ";
    cin >> key;

    int ans = Linear_Search(Array, key);

if (ans != -1) {
        cout << key << " Found at Index - " << ans << " of Array." << endl;
    }
    else {
        cout << "Key is not exists in Array.";
    }
    return 0;
}
```

Output :-



```
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 2> cd "d:\Arya
n Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 2\" ; if ($?) { g++ Line
ar_Search.cpp -o Linear_Search } ; if ($?) { .\Linear_Search }
Enter The Size of the Array :-
10
Enter The Element for the Array:-
Enter Element at index 0 : 23
Enter Element at index 1 : 45
Enter Element at index 2 : 67
Enter Element at index 3 : 89
Enter Element at index 4 : 90
Enter Element at index 5 : 23
Enter Element at index 6 : 17
Enter Element at index 7 : 65
Enter Element at index 8 : 39
Enter Element at index 9 : 71
Your Input Array Is :-
23 45 67 89 90 23 17 65 39 71
Enter the Key to Search In Array :- 23
23 Found at Index - 0 of Array.
PS D:\Aryan Data\Usefull Data\Semester - 5\Semester-5\Design And Analysis of Algorithms\Lab - Manual\Experiment - 2>
```

Space Complexity:- ______________________

Justification: -

____

____

____

____

____

Time Complexity:

Best Case Time Complexity: ___________________

Justification: -

____

____

____

____

____

Worst Case Time Complexity:- ________________

Justification: -

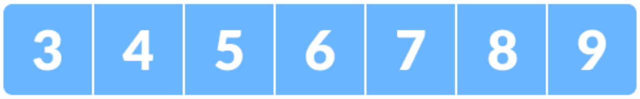
Binary Search

Theory: -

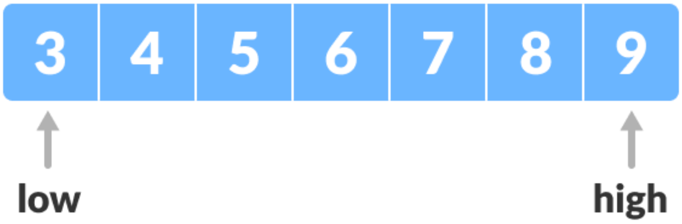
- Binary Search is a searching algorithm for finding an element's position in a sorted array.
- In this approach, the element is always searched in the middle of a portion of an array.
- Binary search can be implemented only on a sorted list of items. If the elements are not sorted already, we need to sort them first.

Working of Bubble Sort

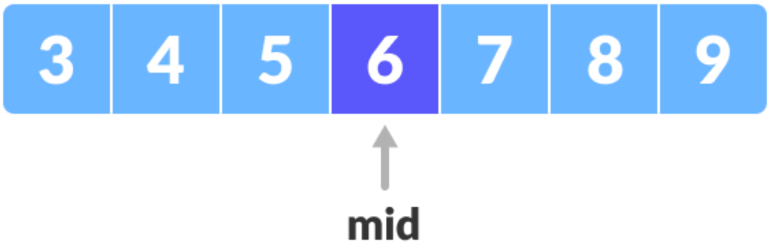
1) Binary Search Algorithm can be implemented using Recursion using divide and conquer approach.
2) The array in which searching is to be performed is: let x = 4 be the element to be searched.



3) Set two pointers low and high at the lowest and the highest positions respectively.



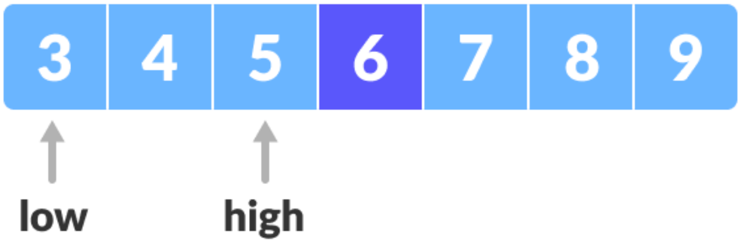
4) Find the middle element mid of the array ie. arr[(low + high)/2] = 6.



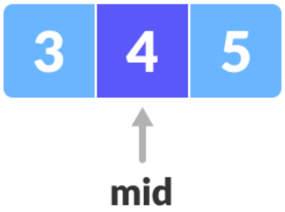
5) If x == mid, then return mid. Otherwise, compare the elements to be searched for with m.

6) If x > mid, compare x with the middle element of the elements on the right side of mid. This is done by setting low to low = mid + 1.

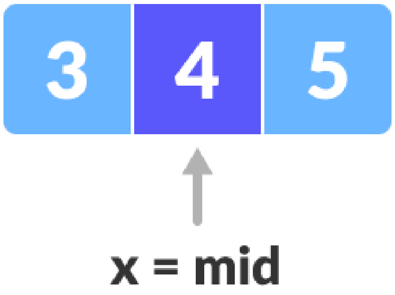
7) Else, compare x with the middle element of the elements on the left side of mid. This is done by setting high to high = mid - 1.



8) Repeat steps 4 to 7 until low meets high.



9) x = 4 is found.



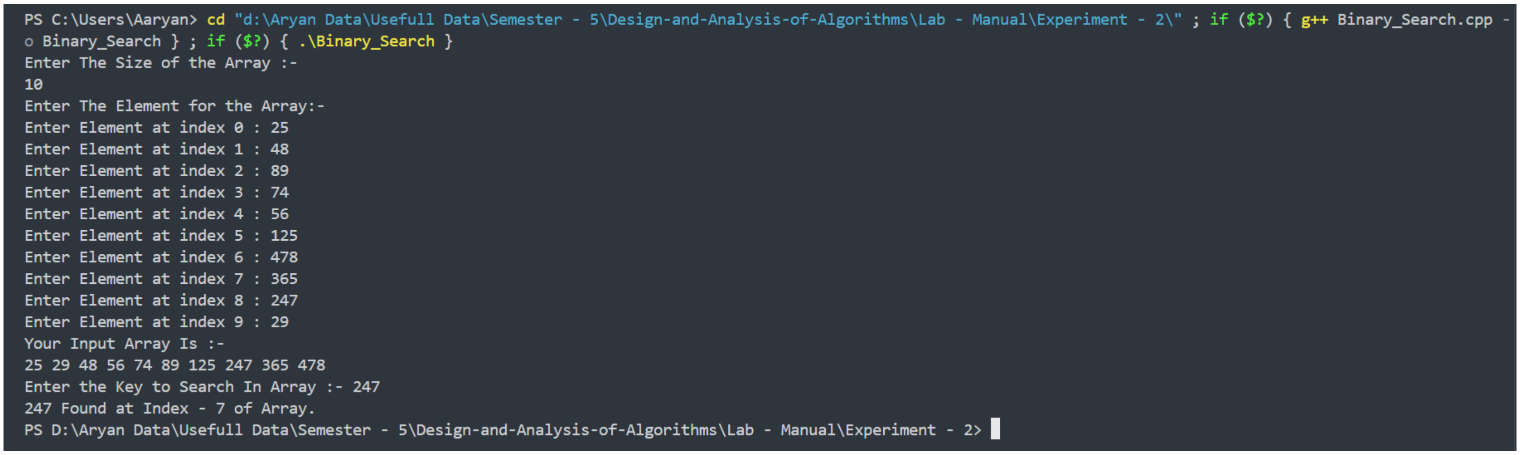
Algorithm: -

Programming Language: - C++

Code :-

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

int Binary_Search(vector<int> Array, int left, int right, int key) {

    if (left <= right) {
        int mid = left + (right - left) / 2;
        if (key == Array[mid]) {
            return mid;
        }

        if (key < Array[mid]) {
            return Binary_Search(Array, left, mid - 1, key);
        }

        else {
            return Binary_Search(Array, mid + 1, right, key);
        }
    }
}

void Print_Array(vector<int> Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << Array[i] << " ";
    }
    cout << endl;
}

void Input_Array(vector<int>& Array) {
    for (int i = 0; i < Array.size(); i++) {
        cout << "Enter Element at index " << i << " : ";
        cin >> Array[i];
    }
}

int main() {
    int size;
    int key;

    while (true) {
        cout << "Enter The Size of the Array :- " << endl;
        cin >> size;

        if (size >= 1) {
            break;
        }

        cout << "Invalid Size. Size must be a Positive Integer." << endl;
    }

    vector<int> Array(size, 0);
    cout << "Enter The Element for the Array:- " << endl;
    Input_Array(Array);
    sort(Array.begin(), Array.end());
    cout << "Your Input Array Is :- " << endl;
    Print_Array(Array);

    cout << "Enter the Key to Search In Array :- ";
    cin >> key;

    int ans = Binary_Search(Array, 0, size, key);

    if (ans != -1) {
        cout << key << " Found at Index - " << ans << " of Array." << endl;
    }

    else {
        cout << "Key is not exists in Array.";
    }
    return 0;
}
```

Output:-



```
PS C:\Users\Aaryan> cd "d:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 2\" ; if ($?) { g++ Binary_Search.cpp -o Binary_Search } ; if ($?) { .\Binary_Search }
Enter The Size of the Array :-
10
Enter The Element for the Array:-
Enter Element at index 0 : 25
Enter Element at index 1 : 48
Enter Element at index 2 : 89
Enter Element at index 3 : 74
Enter Element at index 4 : 56
Enter Element at index 5 : 125
Enter Element at index 6 : 478
Enter Element at index 7 : 365
Enter Element at index 8 : 247
Enter Element at index 9 : 29
Your Input Array Is :-
25 29 48 56 74 89 125 247 365 478
Enter the Key to Search In Array :- 247
247 Found at Index - 7 of Array.
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 2>
```

Space Complexity:- ______________________

Justification: -

____

____

____

____

____

Time Complexity:

Best Case Time Complexity: ___________________

Justification: -

Worst Case Time Complexity:- _________________

Justification: -

Conclusion:-