EXPERIMENT NUMBER- 01

STUDENT'S NAME	Priyanshu Suryawanshi
STUDENT'S UID	21CBS1049
CLASS AND GROUP	21CSB-118
SEMESTER	Second

TOPIC OF EXPERIMENT: Linear Search & Binary Search

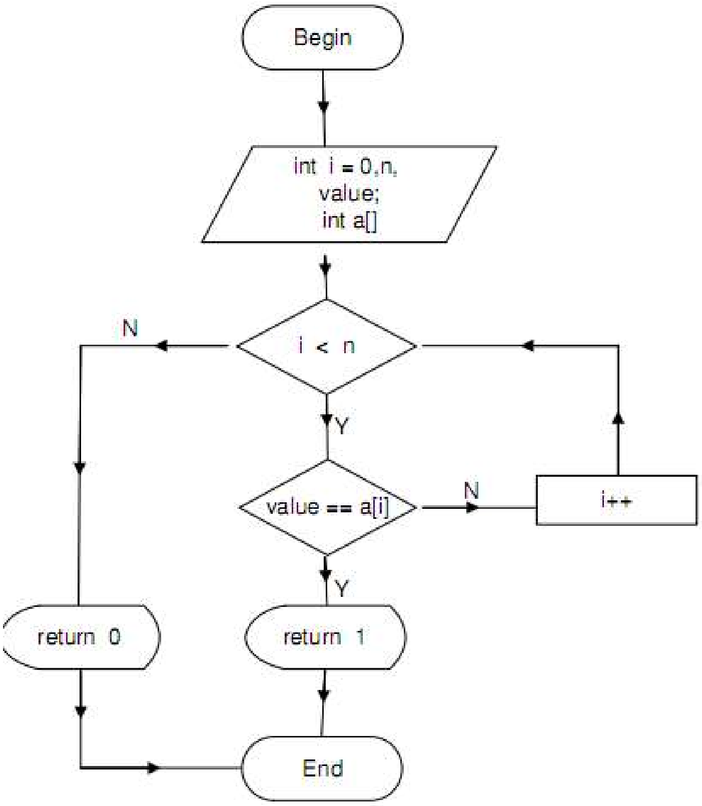
AIM OF THE EXPERIMENT:

→1. Program to demonstrate the use of linear search to search a given element in an array.

→2. Program to demonstrate the use of binary search to search a given element in a sorted array in ascending order.

FLOWCHART/ALGORITHM:

Flowchart for Linear search: -



Algorithm for Program 1:

Linear Search(A,item,location,UB)

Where A is an Array, item is the element to be found .

Location is location of the element

UB is Upper Bound

```
1. Location=0.
2.  Repeat 3 till Location <= UB
3. If A[Location] = item then print location and then break.
else , set location = location  + 1.
4. If (Location = UB+1), location = NULL
5. Exit
```

Algorithm for Program 2:

Binary Search(A,item,location,UB,LB)

Where A is an Array, item is the element to be found .

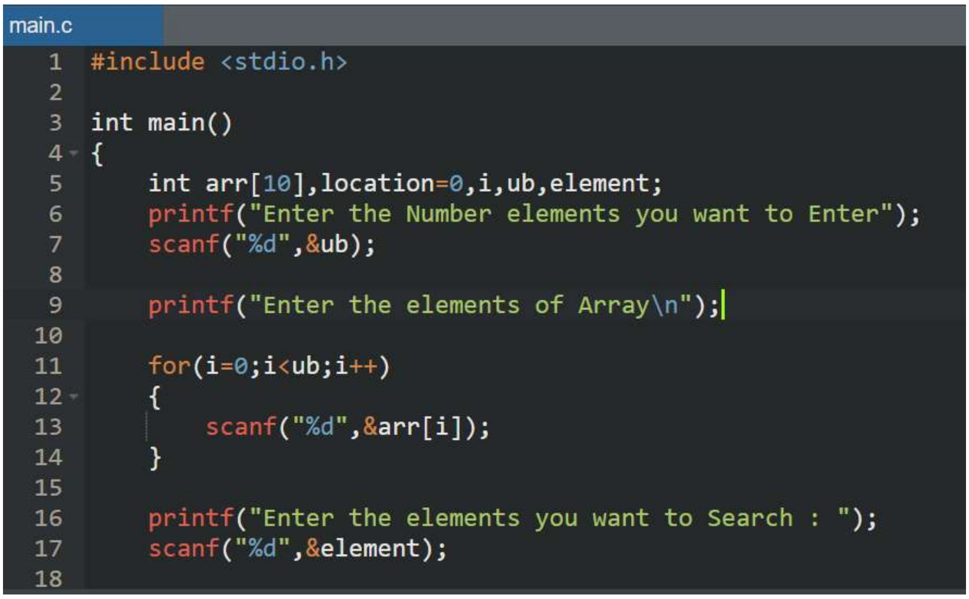
Location is location of the element

UB is Upper Bound and LB is Lower Bound

```
    1. First = LB, Last = UB , Mid = (UB+LB)/2
    2. Repeat 3 and 4 till First <= Last and A[Mid] != item
    3. If item < A[Mid] then Last = Mid – 1
       Else, First = Mid – 1
    4. Set Mid = int((First+Last)/2)
    5. If A[Mid] = item then location = Mid
Else location = NULL
    6. Exit
```

PROGRAM CODE:

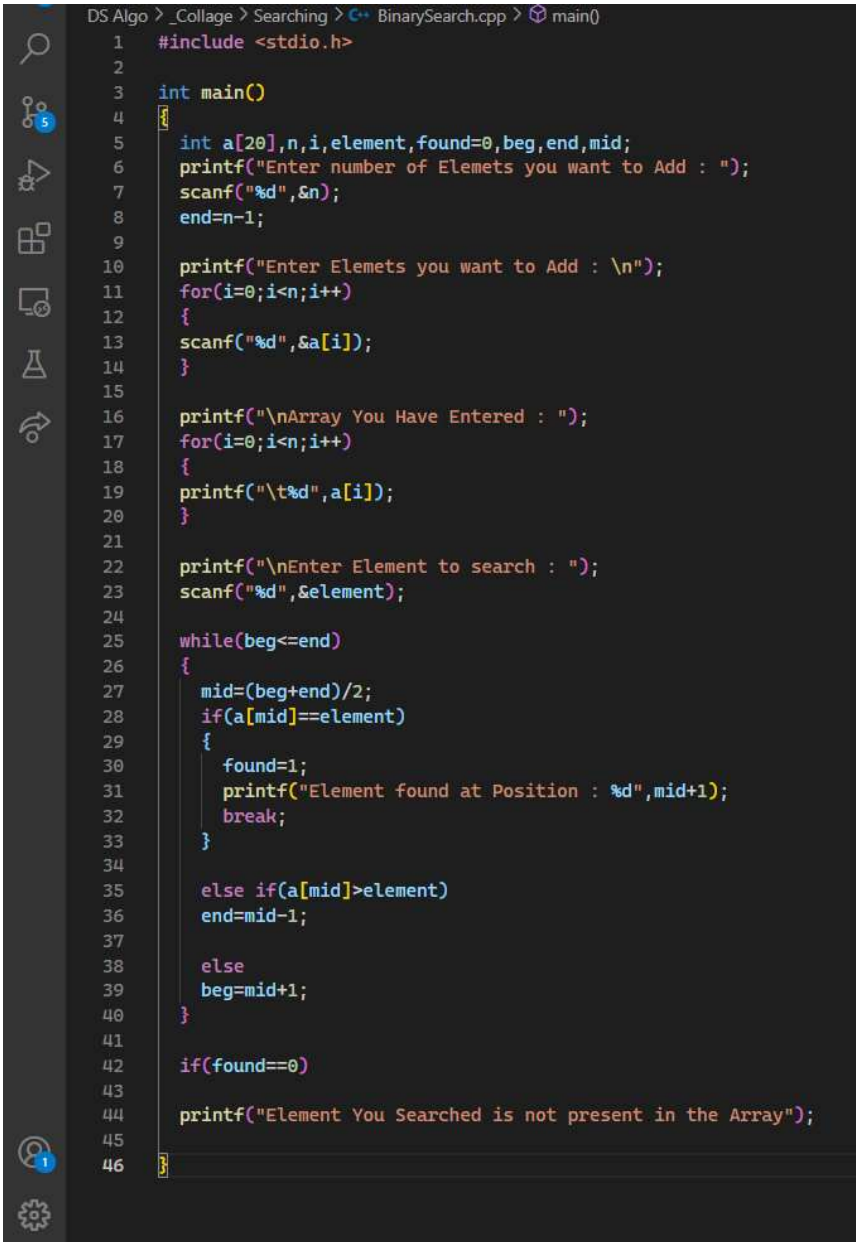
Code for Program 1: -



main.c

```c
#include <stdio.h>

int main()
{
    int arr[10],location=0,i,ub,element;
    printf("Enter the Number elements you want to Enter");
    scanf("%d",&ub);

    printf("Enter the elements of Array\n");

    for(i=0;i<ub;i++)
    {
        scanf("%d",&arr[i]);
    }

    printf("Enter the elements you want to Search : ");
    scanf("%d",&element);

```

Code for Program 2: -



```
#include <stdio.h>

int main()
{
    int a[20],n,i,element,found=0,beg,end,mid;
    printf("Enter number of Elemets you want to Add : ");
    scanf("%d",&n);
    end=n-1;

    printf("Enter Elemets you want to Add : \n");
    for(i=0;i<n;i++)
    {
    scanf("%d",&a[i]);
    }

    printf("\nArray You Have Entered : ");
    for(i=0;i<n;i++)
    {
    printf("\t%d",a[i]);
    }

    printf("\nEnter Element to search : ");
    scanf("%d",&element);

    while(beg<=end)
    {
        mid=(beg+end)/2;
        if(a[mid]==element)
        {
            found=1;
            printf("Element found at Position : %d",mid+1);
            break;
        }

        else if(a[mid]>element)
        end=mid-1;

        else
        beg=mid+1;
    }

    if(found==0)

    printf("Element You Searched is not present in the Array");

}
```

ERRORS ENCOUNTERED DURING PROGRAM'S EXECUTION:

(Same for Both Programes)

; missing

} missing

PROGRAMS' EXPLANATION (in brief)

A linear search scans one item at a time, without jumping to any item .

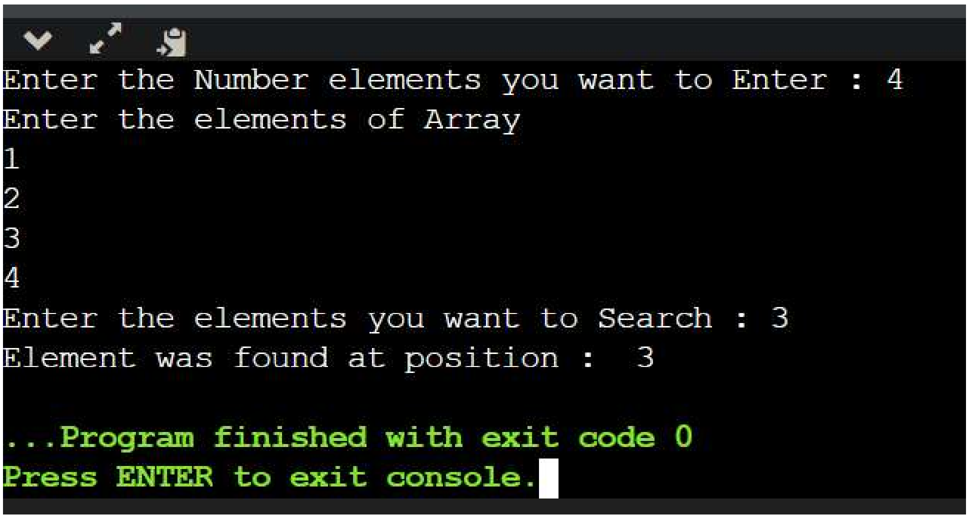
1. The worst case complexity is O(n), sometimes known an O(n) search
2. Time taken to search elements keep increasing as the number of elements are increased.

A binary search however, cut down your search to half as soon as you find middle of a sorted list.

1. The middle element is looked to check if it is greater than or less than the value to be searched.
2. Accordingly, search is done to either half of the given list

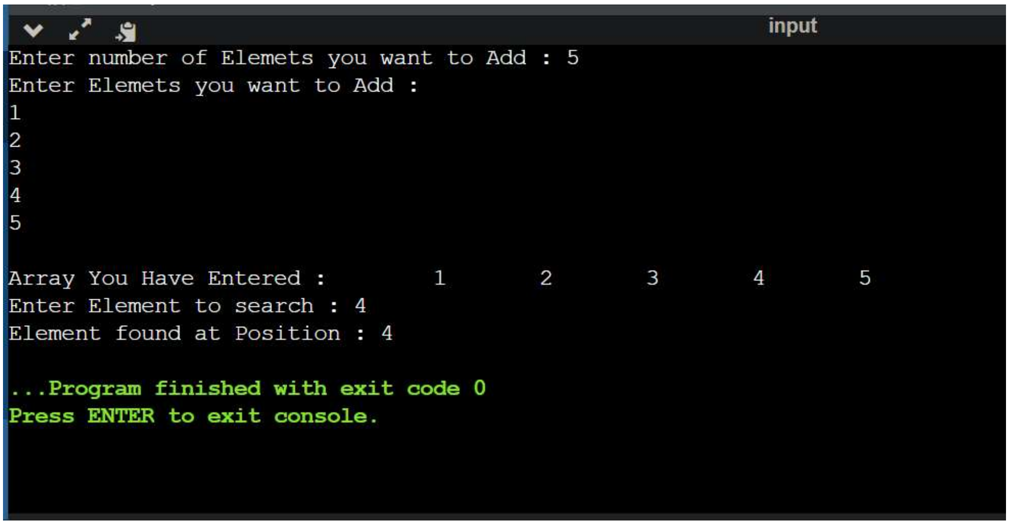
OUTPUT

Output of program 1: -



```
Enter the Number elements you want to Enter : 4
Enter the elements of Array
1
2
3
4
Enter the elements you want to Search : 3
Element was found at position :  3

...Program finished with exit code 0
Press ENTER to exit console.
```

Output of program 2: -



```
Enter number of Elemets you want to Add : 5
Enter Elemets you want to Add :
1
2
3
4
5

Array You Have Entered :        1       2       3       4       5
Enter Element to search : 4
Element found at Position : 4

...Program finished with exit code 0
Press ENTER to exit console.
```

LEARNING OUTCOMES

- Analyze and compare the efficiency and properties of various data structures.
- Identify strength and weaknesses of various data structures.
- Design and employ appropriate data structures for solving computing problems.
- Possess the ability to design efficient algorithms for solving computing problems.

EVALUATION COLUMN (To be filled by concerned faculty only)

Sr. No.	Parameters	Maximum Marks	Marks Obtained
1.	Prelab questions	5	
2.	Completion of worksheet with learning outcomes and program's output along with cleanliness and discipline.	10	
3.	Post lab Questions	5	
4.	Total Marks	20	
5.	Teacher's Signature (with date)		