NAME: ADITYA ABHIJIT PARULEKAR

DIV: S2-1

ROLL.NO: 2201072

Experiment 3:

- **AIM:** To study and implement Binary search algorithm.

- **THEORY:**

Binary Search Algorithm: Binary search is the search technique that works efficiently on sorted lists. Hence, to search an element into some list using the binary search technique, we must ensure that the list is sorted.

Binary search follows the divide and conquer approach in which the list is divided into two halves, and the item is compared with the middle element of the list. If the match is found then, the location of the middle element is returned. Otherwise, we search into either of the halves depending upon the result produced through the match.

Algorithm

```
Binary_Search(a, lower_bound, upper_bound, val) // 'a' is the given array, 'lower_bound' is the index of
Step 1: set beg = lower_bound, end = upper_bound, pos = - 1
Step 2: repeat steps 3 and 4 while beg <=end
Step 3: set mid = (beg + end)/2
Step 4: if a[mid] = val
set pos = mid
print pos
go to step 6
else if a[mid] > val
set end = mid - 1
else
set beg = mid + 1
[end of if]
[end of loop]
Step 5: if pos = -1
print "value is not present in the array"
[end of if]
Step 6: exit
```

Working of Binary search:

Now, let's see the working of the Binary Search Algorithm.

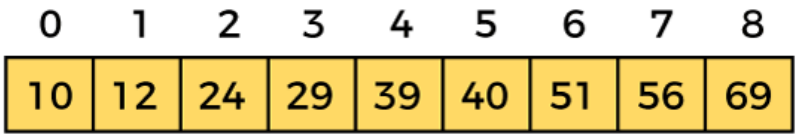
To understand the working of the Binary search algorithm, let's take a sorted array. It will be easy to understand the working of Binary search with an example.

There are two methods to implement the binary search algorithm –

- Iterative method
- Recursive method

The recursive method of binary search follows the divide and conquer approach.

Let the elements of array are –



Let the element to search is, **K = 56**

We have to use the below formula to calculate the **mid** of the array -

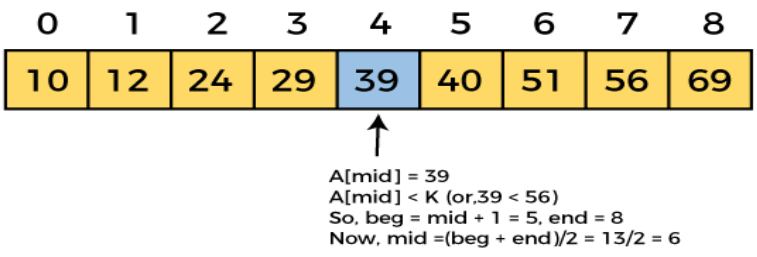
$$mid = (beg + end)/2$$

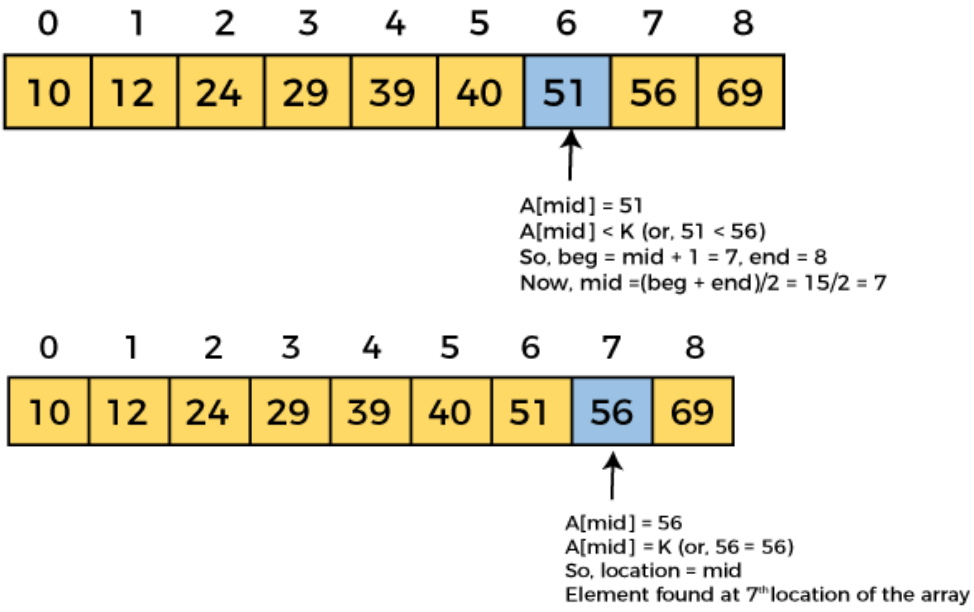
So, in the given array -

beg = 0

end = 8

mid = (0 + 8)/2 = 4. So, 4 is the mid of the array.





Now, the element to search is found. So algorithm will return the index of the element matched.

➢ Binary Search complexity:

Now, let's see the time complexity of Binary search in the best case, average case, and worst case. We will also see the space complexity of Binary search.

1. Time Complexity:

Case	Time Complexity
Best Case	O(1)
Average Case	O(logn)
Worst Case	O(logn)

- **Best Case Complexity -** In Binary search, best case occurs when the element to search is found in first comparison, i.e., when the first middle element itself is the element to be searched. The best-case time complexity of Binary search is **O(1).**
- **Average Case Complexity -** The average case time complexity of Binary search is **O(logn).**
- **Worst Case Complexity -** In Binary search, the worst case occurs, when we have to keep reducing the search space till it has only one element. The worst-case time complexity of Binary search is **O(logn).**

2. Space Complexity:

Space Complexity	O(1)

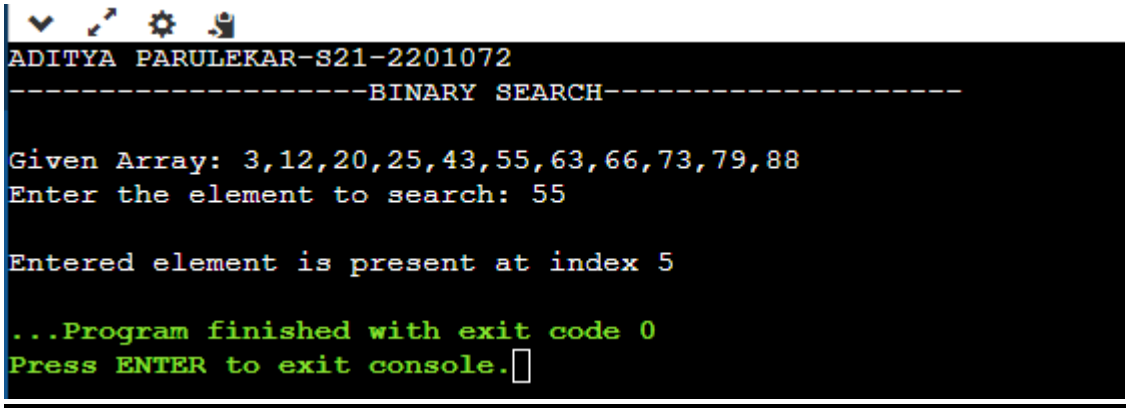
- The space complexity of binary search is O(1).

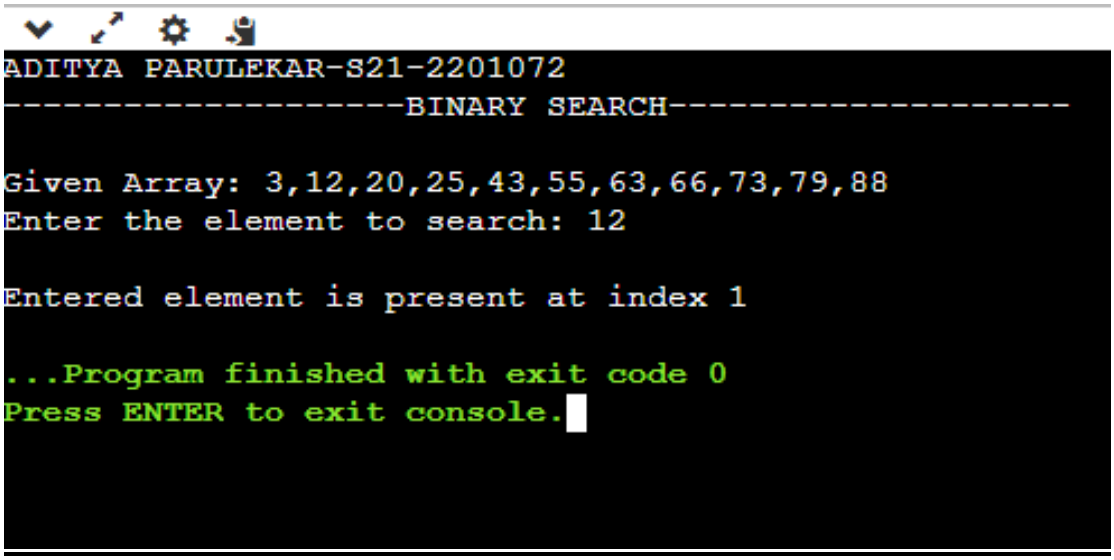
- **C PROGRAM:**

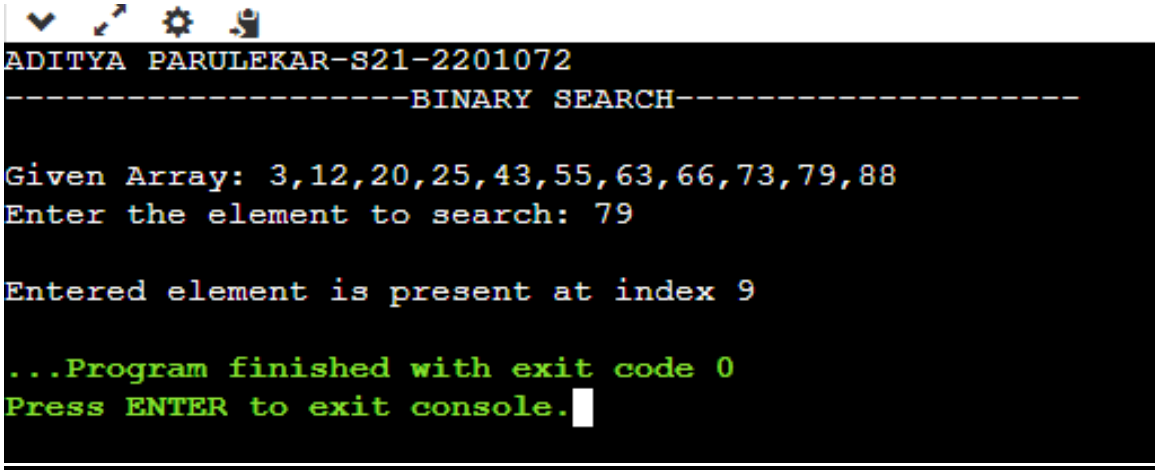
```
#include <stdio.h>

int binarySearch(int arr[], int l, int r, int x )
{
   if(r>=l)
   {
      int m = l + (r-l) / 2;

      if(arr[m]==x)
      return m;

      if(arr[m]>x)
      return binarySearch(arr, l, m-1, x);

      else
      return binarySearch(arr, m+1, r, x);
   }
   return -1;
}
```

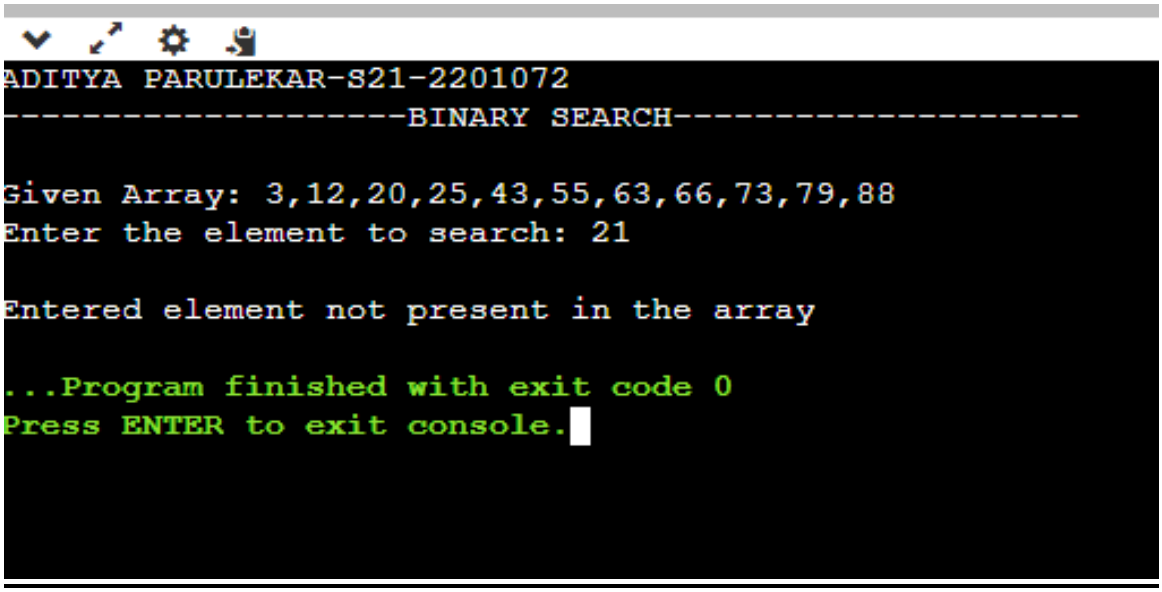
```
int main()
{
   int x;
   printf("ADITYA PARULEKAR-S21-2201072");
   printf("\n--------------------BINARY SEARCH--------------------\n\n");
   int arr[] = {3,12,20,25,43,55,63,66,73,79,88};

   int n = sizeof(arr)/sizeof(arr[0]);

   printf("Given Array: ");
   for(int i=0; i<n-1; i++)
   {
      printf("%d,", arr[i]);
   }
   printf("%d", arr[n-1]);

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

   int result = binarySearch(arr, 0, n-1, x);

   (result == -1) ? printf("\nEntered element not present in the array") :
printf("\nEntered element is present at index %d", result);
   return 0;

}
```

- **OUTPUT:**


ADITYA PARULEKAR-S21-2201072
--------------------BINARY SEARCH--------------------
Given Array: 3,12,20,25,43,55,63,66,73,79,88
Enter the element to search: 55
Entered element is present at index 5
...Program finished with exit code 0
Press ENTER to exit console.


ADITYA PARULEKAR-S21-2201072
--------------------BINARY SEARCH--------------------
Given Array: 3,12,20,25,43,55,63,66,73,79,88
Enter the element to search: 12
Entered element is present at index 1
...Program finished with exit code 0
Press ENTER to exit console.


ADITYA PARULEKAR-S21-2201072
--------------------BINARY SEARCH--------------------
Given Array: 3,12,20,25,43,55,63,66,73,79,88
Enter the element to search: 79
Entered element is present at index 9
...Program finished with exit code 0
Press ENTER to exit console.



CONCLUSION: Hence, we have successfully implemented Binary search algorithm; LO1, LO2.