Experiment No. : 08

Title: Implementation of searching algorithm

Batch:A2

Roll No.:16010421063

Aim: To implement binary search algorithm using array.

Resources Used: Turbo C/ C++ editor and C compiler.

Theory:

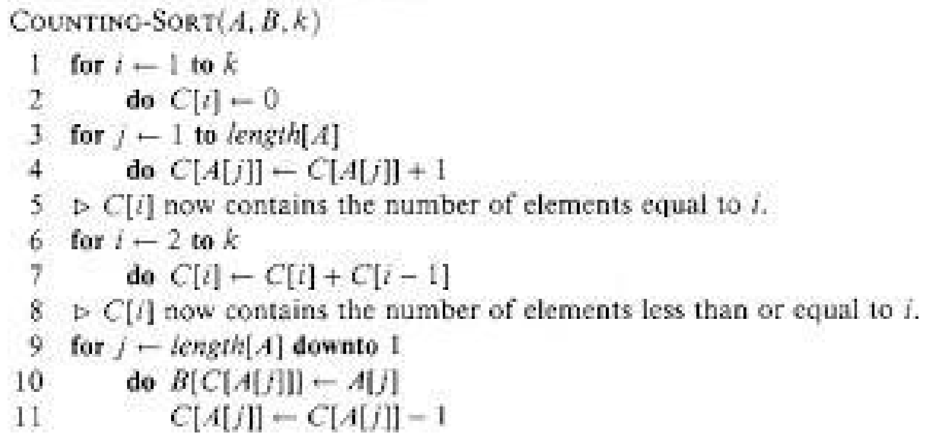
Searching –

Search is a process of finding a value in a list of values. In other words, searching is the process of locating given value position in a list of values.

The **binary search algorithm** can be used with only sorted list of element. That means, binary search can be used only with list of element which are already arranged in a order. The binary search cannot be used for list of element which are in random order. This search process starts comparing of the search element with the middle element in the list. If both are matched, then the result is "element found". Otherwise, we check whether the search element is smaller or larger than the middle element in the list. If the search element is smaller, then we repeat the same process for left sub-list of the middle element. If the search element is larger, then we repeat the same process for right sub-list of the middle element. We repeat this process until we find the search element in the list or until we left with a sub-list of only one element. And if that element also doesn't match with the search element, then the result is "Element not found in the list".

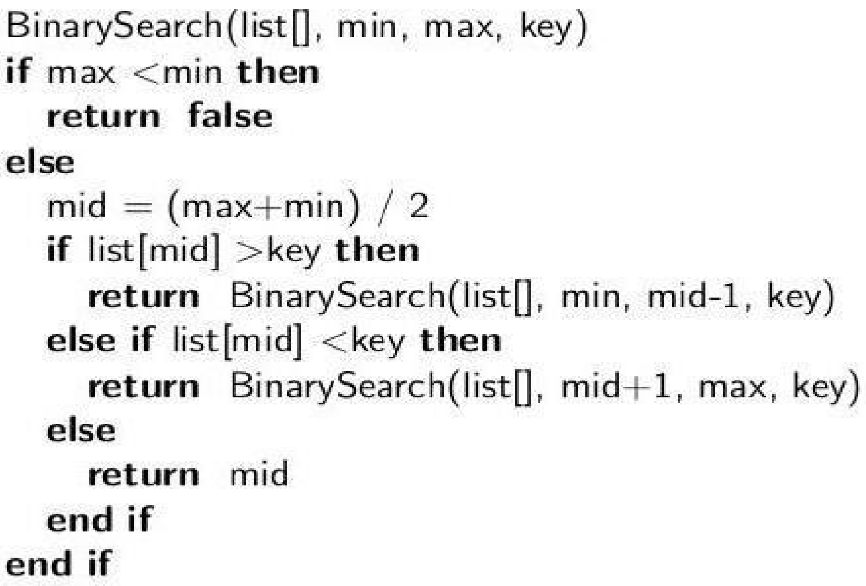
Algorithm :

1) **PreCondition -** Sort the given input array using counting sort**.**



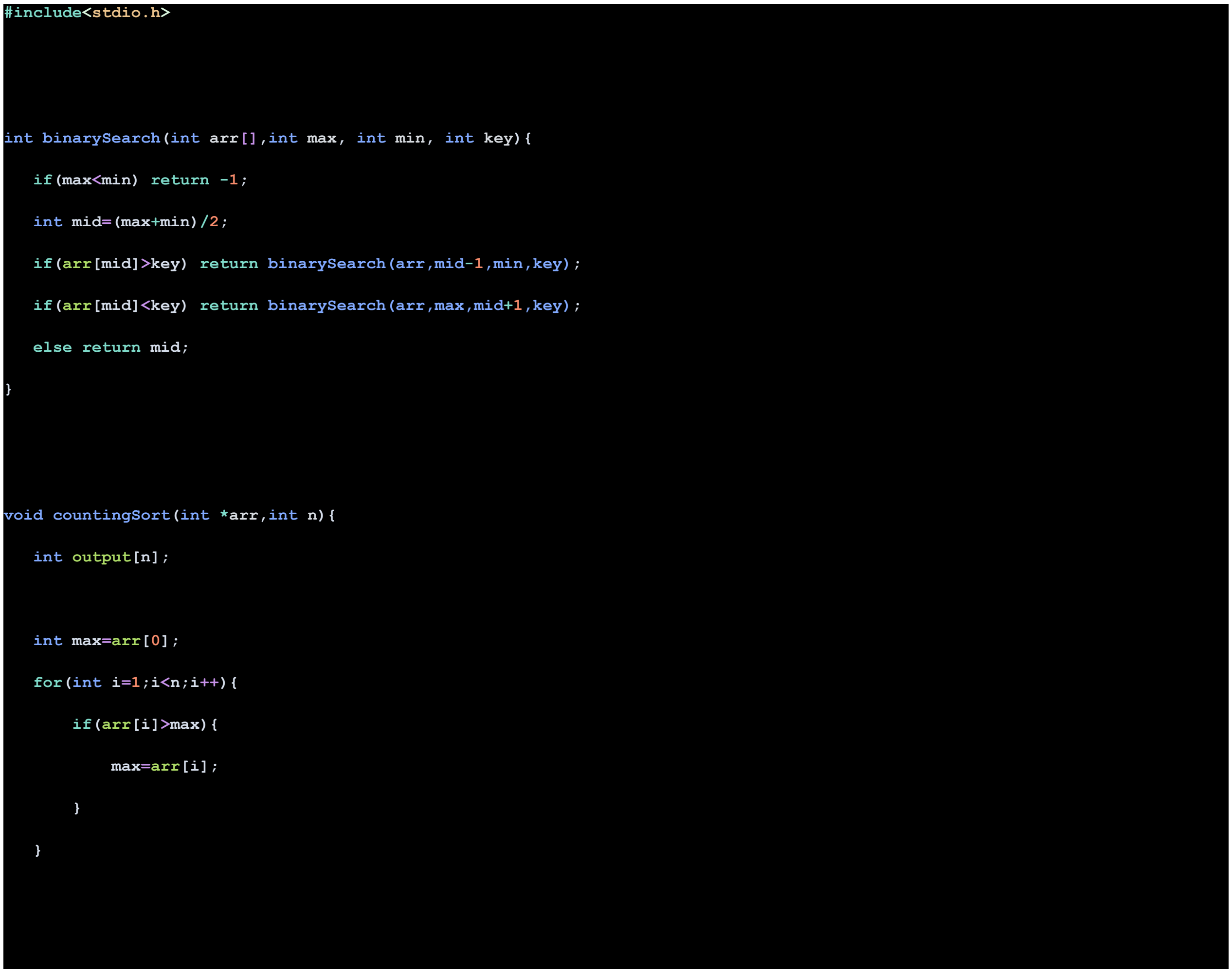
```
COUNTING-SORT(A, B, k)
for i ← 1 to k
    do C[i] ← 0
for j ← 1 to length[A]
    do C[A[j]] ← C[A[j]] + 1
▷ C[i] now contains the number of elements equal to i.
for i ← 2 to k
    do C[i] ← C[i] + C[i − 1]
▷ C[i] now contains the number of elements less than or equal to i.
for j ← length[A] downto 1
    do B[C[A[j]]] ← A[j]
       C[A[j]] ← C[A[j]] − 1
```

2) **Binary Search algorithm -**

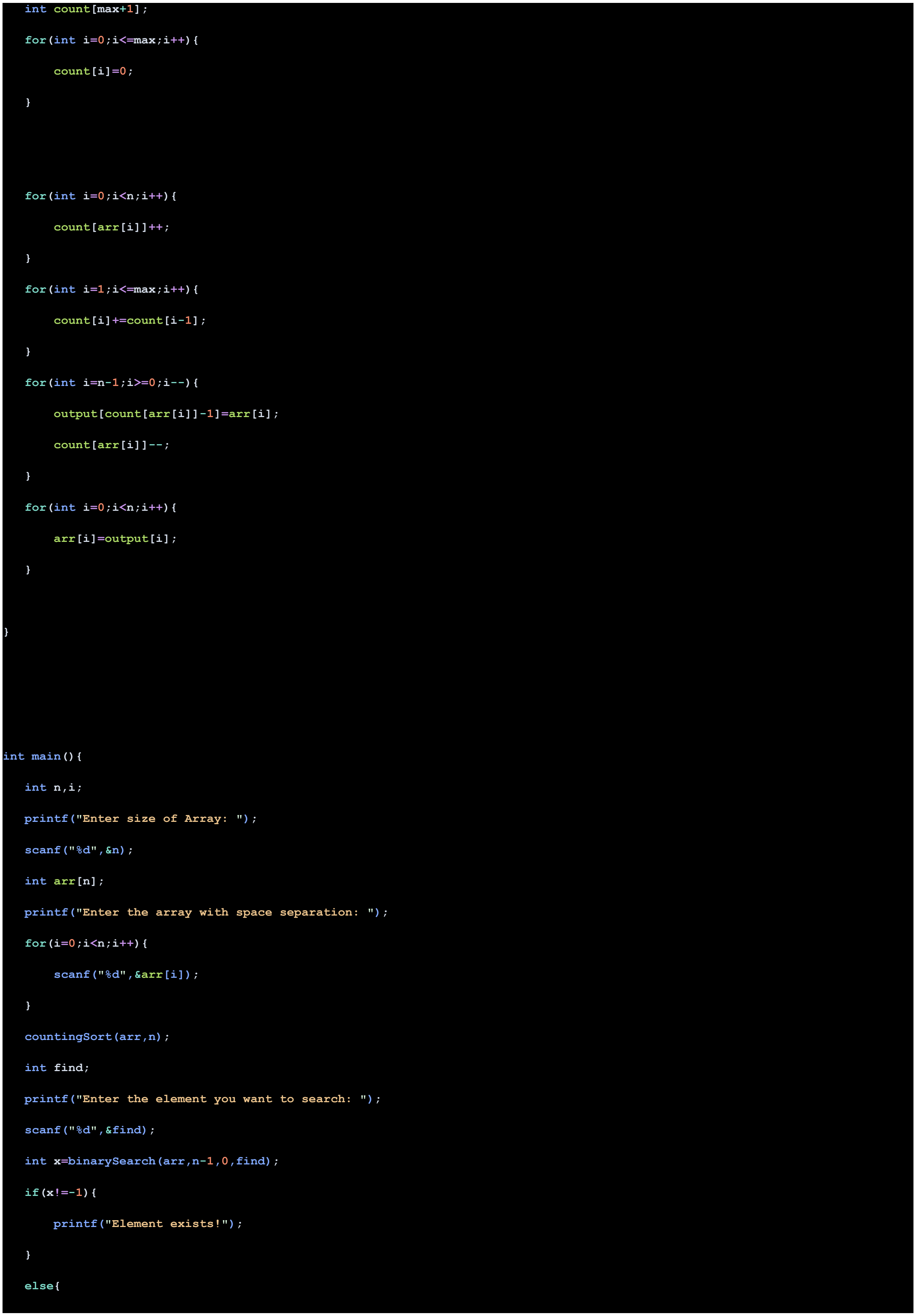


```
BinarySearch(list[], min, max, key)
if max <min then
    return false
else
    mid = (max+min) / 2
    if list[mid] >key then
        return BinarySearch(list[], min, mid-1, key)
    else if list[mid] <key then
        return BinarySearch(list[], mid+1, max, key)
    else
        return mid
    end if
end if
```

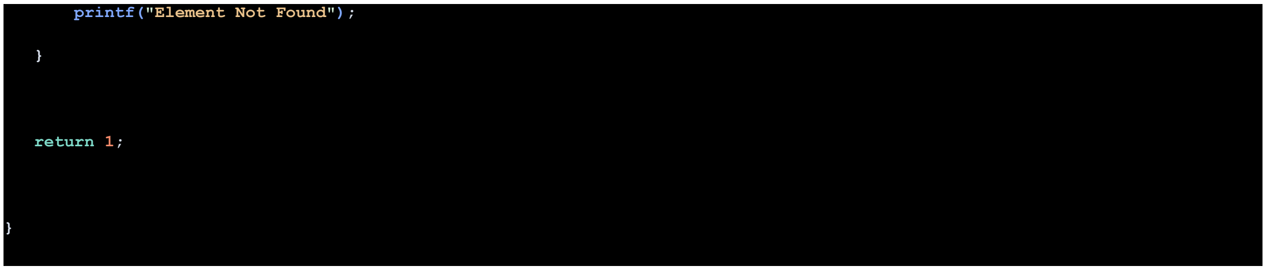
Program



```
#include<stdio.h>


int binarySearch(int arr[],int max, int min, int key){
   if(max<min) return -1;
   int mid=(max+min)/2;
   if(arr[mid]>key) return binarySearch(arr,mid-1,min,key);
   if(arr[mid]<key) return binarySearch(arr,max,mid+1,key);
   else return mid;
}


void countingSort(int *arr,int n){
   int output[n];

   int max=arr[0];
   for(int i=1;i<n;i++){
       if(arr[i]>max){
           max=arr[i];
       }
   }
```

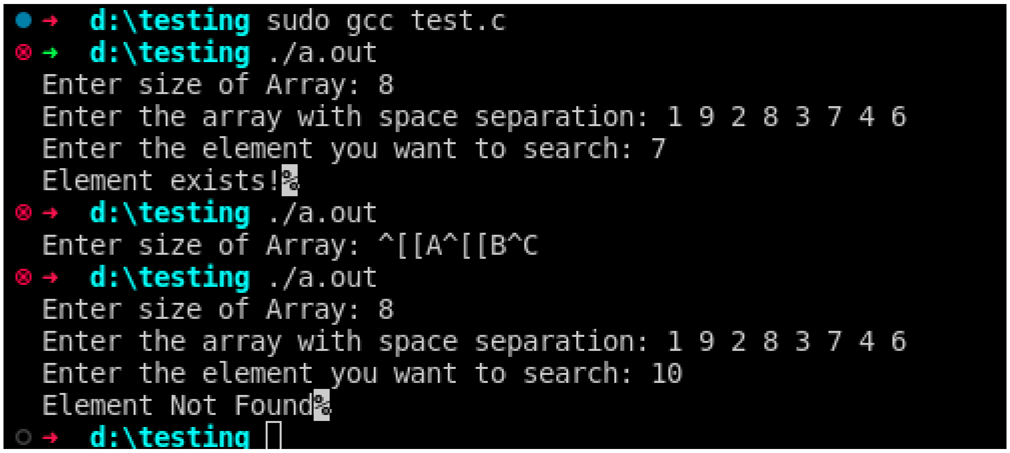


```
    int count[max+1];
    for(int i=0;i<=max;i++){
        count[i]=0;
    }



    for(int i=0;i<n;i++){
        count[arr[i]]++;
    }
    for(int i=1;i<=max;i++){
        count[i]+=count[i-1];
    }
    for(int i=n-1;i>=0;i--){
        output[count[arr[i]]-1]=arr[i];
        count[arr[i]]--;
    }
    for(int i=0;i<n;i++){
        arr[i]=output[i];
    }

}



int main(){
    int n,i;
    printf("Enter size of Array: ");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the array with space separation: ");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    countingSort(arr,n);
    int find;
    printf("Enter the element you want to search: ");
    scanf("%d",&find);
    int x=binarySearch(arr,n-1,0,find);
    if(x!=-1){
        printf("Element exists!");
    }
    else{
```



```
        printf("Element Not Found");

    }



    return 1;



}
```

OUTPUT



```
● → d:\testing sudo gcc test.c
⊗ → d:\testing ./a.out
  Enter size of Array: 8
  Enter the array with space separation: 1 9 2 8 3 7 4 6
  Enter the element you want to search: 7
  Element exists!%
⊗ → d:\testing ./a.out
  Enter size of Array: ^[[A^[[B^C
⊗ → d:\testing ./a.out
  Enter size of Array: 8
  Enter the array with space separation: 1 9 2 8 3 7 4 6
  Enter the element you want to search: 10
  Element Not Found%
○ → d:\testing
```

Outcomes:

CO4: Demonstrate Sorting and Searching methods

Conclusion-

Understood the algorithm of Counting sort and Binary Search and implemented the same

References:

Books/ Journals/ Websites:

- Y. Langsam, M. Augenstin and A. Tenenbaum, "Data Structures using C", Pearson Education Asia, 1st Edition, 2002.
- Vlabs on binary search and counting sort.