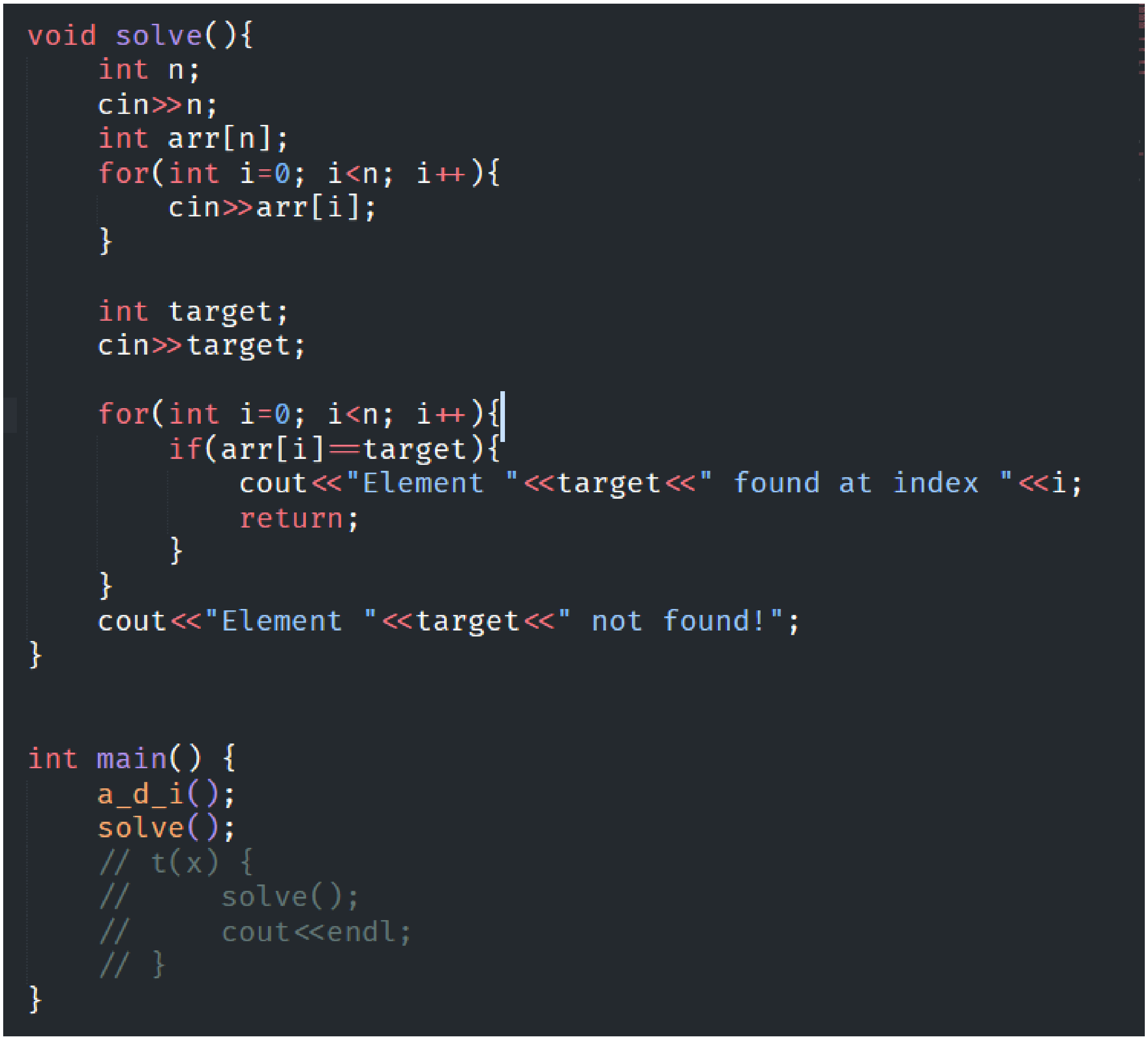
IT DS 201 LAB

LAB 1 (SEARCHING)

SUBMITTED BY ADITYA SINGH 2K19/EP/005

Program 1: Write a program to implement Linear Search

CODE



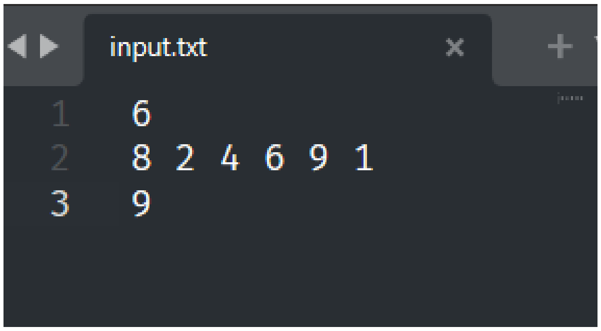
```
void solve(){
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    int target;
    cin>>target;

    for(int i=0; i<n; i++){
        if(arr[i]==target){
            cout<<"Element "<<target<<" found at index "<<i;
            return;
        }
    }
    cout<<"Element "<<target<<" not found!";
}


int main() {
    a_d_i();
    solve();
    // t(x) {
    //     solve();
    //     cout<<endl;
    // }
}
```

ALGORITHM

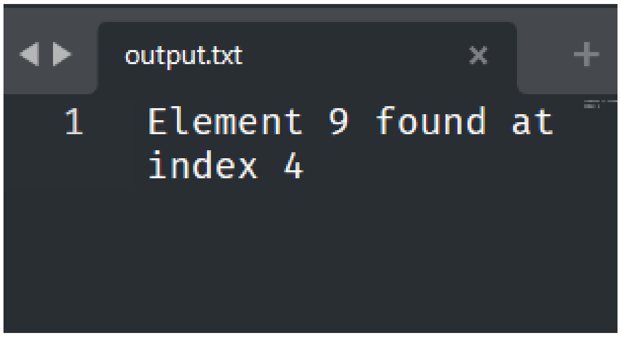
- Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
- If x matches with an element, return the index.
- If x doesn't match with any of the elements, return -1.

INPUT/OUTPUT

1.

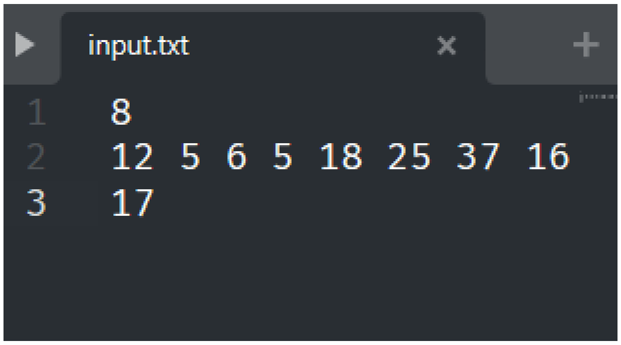


```
6
8 2 4 6 9 1
9
```

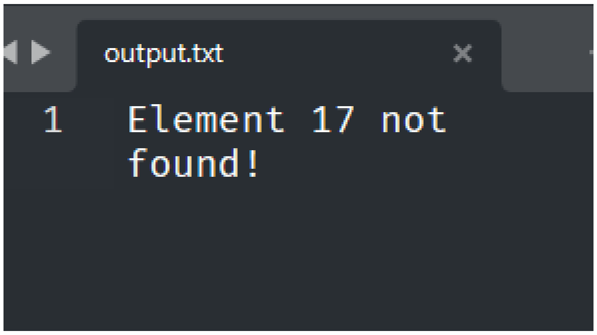


```
Element 9 found at
index 4
```

2.

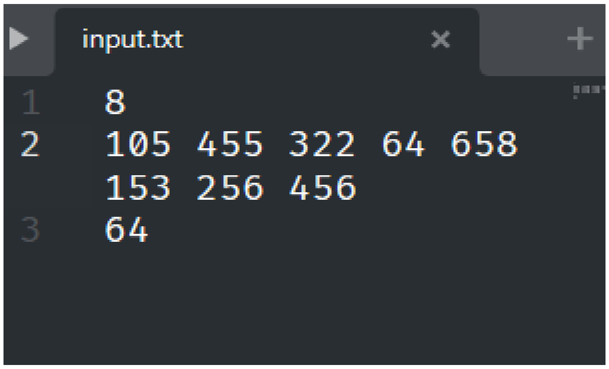


```
8
12 5 6 5 18 25 37 16
17
```

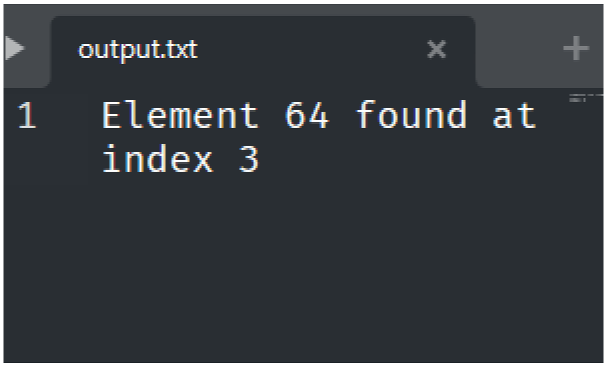


```
Element 17 not
found!
```

3.



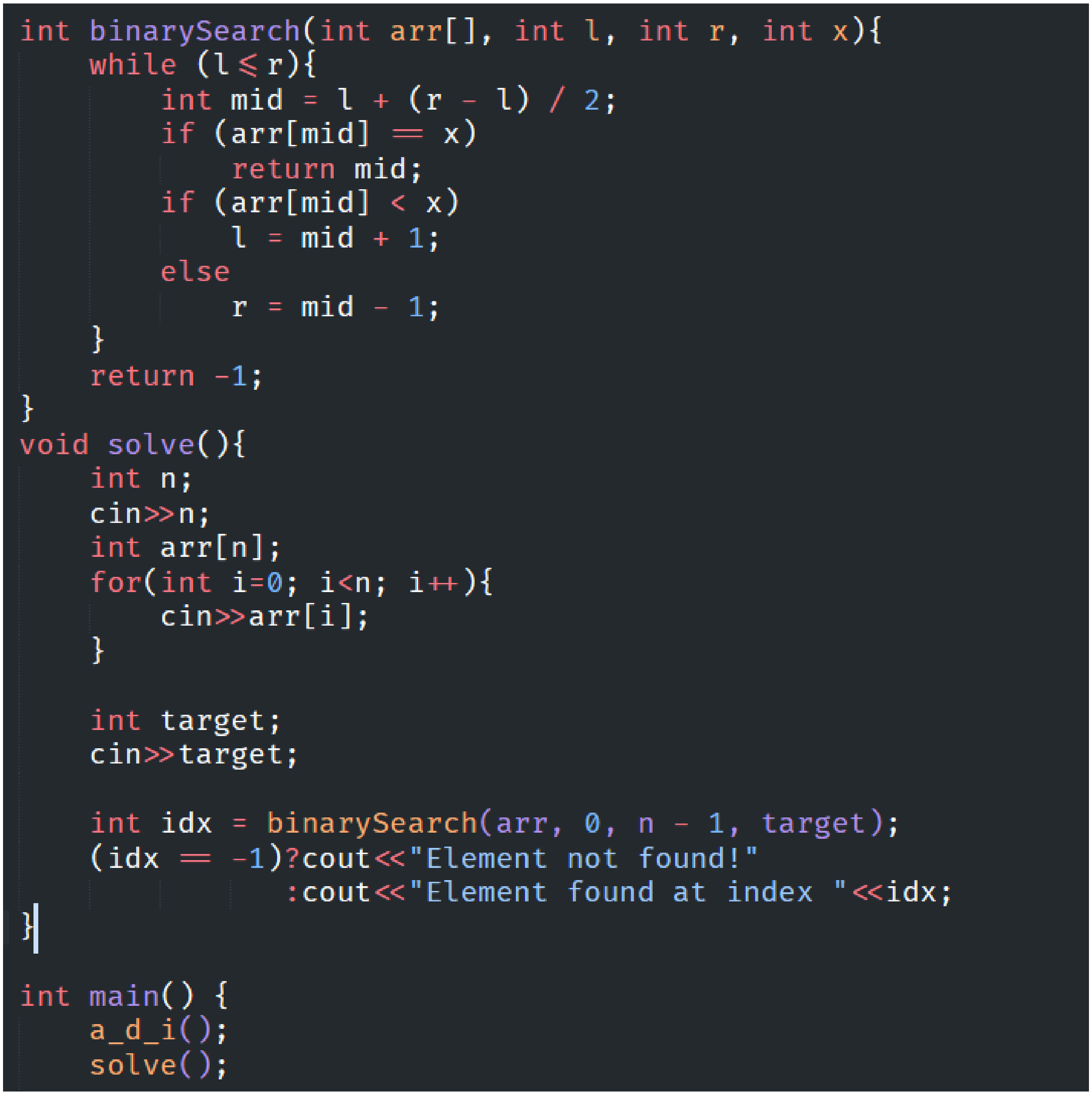
```
8
105 455 322 64 658
153 256 456
64
```



```
Element 64 found at
index 3
```

Program 2: Write a program to implement Binary Search. Assume that the array list is already sorted.

CODE



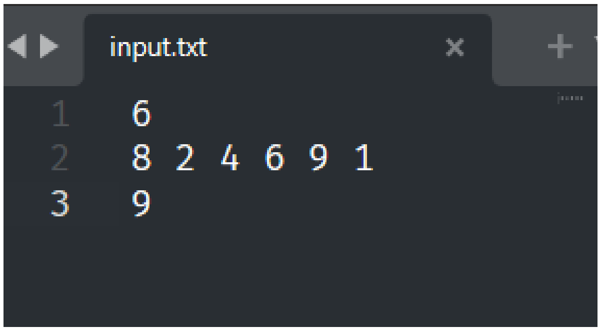
```
int binarySearch(int arr[], int l, int r, int x){
    while (l <= r){
        int mid = l + (r - l) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] < x)
            l = mid + 1;
        else
            r = mid - 1;
    }
    return -1;
}
void solve(){
    int n;
    cin>>n;
    int arr[n];
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    int target;
    cin>>target;

    int idx = binarySearch(arr, 0, n - 1, target);
    (idx == -1)?cout<<"Element not found!"
               :cout<<"Element found at index "<<idx;
}

int main() {
    a_d_i();
    solve();
```

ALGORITHM

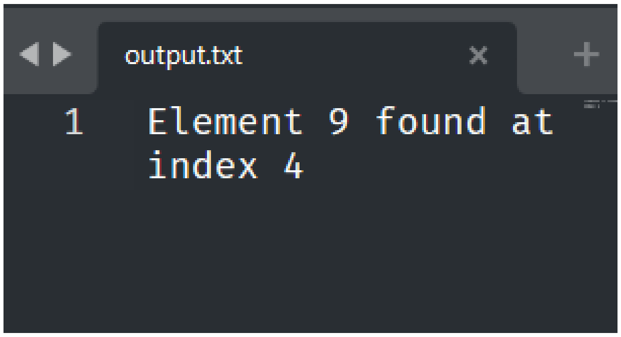
- Compare x with the middle element.
- If x matches with the middle element, we return the mid index.
- Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for the right half.
- Else (x is smaller) recur for the left half.

INPUT/OUTPUT

1.

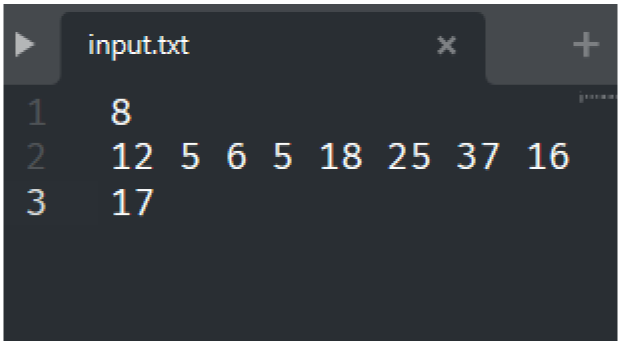


```
6
8 2 4 6 9 1
9
```

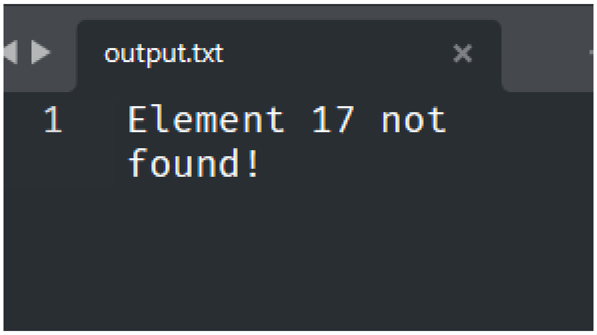


```
Element 9 found at index 4
```

2.

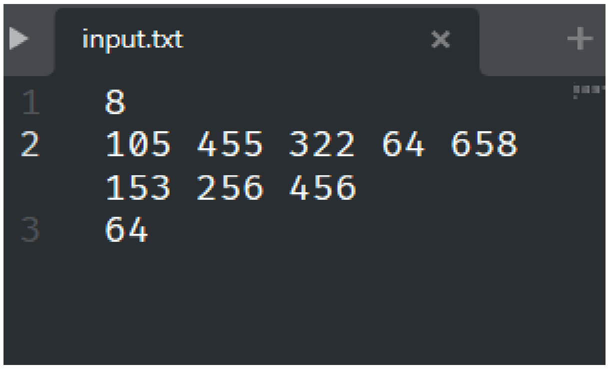


```
8
12 5 6 5 18 25 37 16
17
```

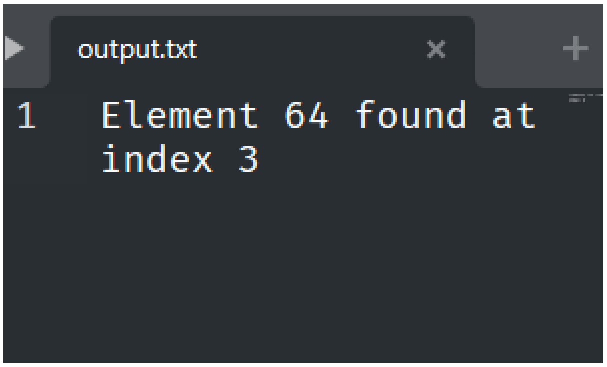


```
Element 17 not found!
```

3.



```
8
105 455 322 64 658 153 256 456
64
```



```
Element 64 found at index 3
```