Harsh Dixit

```
// BINARY SEARCH

/*
   Binary Search can only be applied on sorted array that means it must be
   monotonically increasing or decreasing.

   Why Binay Search?

   In Binay Search after every iteration the size of the array gets reduced to the
   half of before. So, number of operations gets reduced too.

   Whereas in linear search, we have to traverse the whole array.

   Time Complexity of binarySearch is --> O(log n)
 */

#include <iostream>
using namespace std;

bool binarySearch(int arr[] , int size , int target)
{
    int start = 0;
    int end = size - 1;
    int mid = start + (end-start)/2;
    while(start <= end)
    {
        if(arr[mid] == target)
        {
            return true;
        }
        else if(arr[mid] > target)
        {
            end = mid - 1;
        }
        else if(arr[mid] < target)

        {
            start = mid + 1;
        }
        mid = start + (end - start)/2;
    }
    return false;
}

int main()
{
    int arr[] = {10,15,20,22,56};
    int size = 5;
    int target = 20;
    int ans = binarySearch(arr,size,target);
    if(ans == 1)
    {
        cout << "Target Found..";
    }
    else
    {
        cout << "Target not found..";
    }
    return 0;
}

//----------------------------------------------------------------------------------------
---

// binarySearch using vector

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

bool binarySearch(vector<int> &v, int target)
{
    int start = 0;
    int end = v.size() - 1;
    int mid = start + (end-start)/2;
    while(start <= end)
    {
        if(v[mid] == target)
        {
            return true;
        }
        else if(v[mid] > target)
        {
            end = mid - 1;
        }
        else if(v[mid] < target)
        {
            start = mid + 1;
        }
        mid = start + (end - start)/2;
    }
    return false;
}

int main()
{
    vector<int>v = {10,20,30,45,67};
    int target = 20;
    int ans = binarySearch(v,target);
    if(ans == 1)
    {
        cout << "Target Found..";
    }
    else
    {
        cout << "Target not found..";
    }
    return 0;
}

//----------------------------------------------------------------------------------------
---

// Find first occurance of element in an array

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

```
int firstOccurance(int arr[8] , int n , int target)
{
    int start = 0;
    int end = n-1;
    int mid = start + (end-start)/2;
    int ans = -1;

    while(start <= end)
    {
        if(target == arr[mid])
        {
            ans = mid;
            end = mid - 1;
        }
        else if(target > arr[mid])
        {
            start = mid + 1;
        }
        else if(target < arr[mid])
        {
            end = mid -1;
        }
        mid = start + (end - start)/2;
    }
    return ans;
}

int main()
{
    int arr[8] = {10,20,40,40,40,40,80,90};
    int n = 8;
    int target = 40;
    int ansIndex = firstOccurance(arr , n , target);
    if(ansIndex == -1)
    {
        cout << "Element not found....";
    }
    else
    {
        cout<< "Element first occurance found at: " <<ansIndex << endl;
    }
    return 0;
}

//------------------------------------------------------------------------------------------
---

// Find first occurance using vector

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

int firstOccurance(vector<int> &v, int target)
{   int ans = -1;
    int start = 0;
    int end = v.size() - 1;
    int mid = start + (end-start)/2;
    while(start <= end)
    {
        if(v[mid] == target)
        {
            ans = mid;
            end = mid - 1;
        }
        else if(v[mid] > target)
        {
            end = mid - 1;
        }
        else if(v[mid] < target)
        {
            start = mid + 1;
        }
        mid = start + (end - start)/2;
    }
    return ans;
}

int main()
{
    vector<int>v = {10,20,40,40,40,40,80,90};
    int target = 40;
    int ans = firstOccurance(v,target);
    if(ans == -1)
    {
        cout << "Element not present..";
    }
    else
    {
        cout << "First Occurance of " << target << " is: " << ans;
    }
    return 0;
}
//------------------------------------------------------------------------------------------
---

// Find Last Occurance element

#include <iostream>
using namespace std;

int lastOccurance(int arr[8] , int n , int target)
{
    int start = 0;
    int end = n-1;
    int mid = start + (end-start)/2;
    int ans = -1;

    while(start <= end)
    {
        if(target == arr[mid])
        {
            ans = mid;
            start = mid + 1;
        }
        else if(target > arr[mid])
        {
            start = mid + 1;
        }
        else if(target < arr[mid])
        {
            end = mid -1;
        }
        mid = start + (end - start)/2;
    }
    return ans;
}

int main()
{
    int arr[8] = {10,20,40,40,40,40,80,90};
    int n = 8;
    int target = 40;
    int ansIndex = lastOccurance(arr , n , target);
    if(ansIndex == -1)
    {
        cout << "Element not found....";
    }
    else
    {
        cout<< "Element last occurance found at: " <<ansIndex << endl;
    }
    return 0;
}

//------------------------------------------------------------------------------------------
---

// Total occurance of element in sorted array
#include <iostream>
using namespace std;

int firstoccurance(int arr[7] , int size , int target)
{
    int start = 0;
    int end = size -1;
    int mid = start + (end-start)/2;
    int ans = -1;

    while(start <= end)
    {
        if(target == arr[mid])
        {
            ans = mid;
            end = mid -1;
        }
        else if(target > arr[mid])
        {
            start = mid + 1;
        }
        else if(target < arr[mid])
        {
            end = mid -1;
        }
        mid = start + (end - start)/2;
    }
    return ans;
}

int lastoccurance(int arr[7] , int size , int target)
{
    int start = 0;
    int end = size -1;
    int mid = start + (end -  start)/2;
    int ans = -1;

    while(start <= end)
    {
        if(target == arr[mid])
        {
            ans = mid;
            start = mid + 1;
        }
        else if(target > arr[mid])
        {
            start = mid + 1;
        }
        else if (target < arr[mid])
        {
            end =  mid -1;
        }
        mid = start + (end -  start)/2;
    }
    return ans;
}

int main()
{
    int arr[7] = {10,20,30,30,30,30,40};
    int size = 7;
    int target = 30;
    int Ans_1 = firstoccurance(arr , size , target);
    int Ans_2 = lastoccurance(arr , size ,target);
    int TotalOccurance = Ans_2 - Ans_1 + 1;
    cout <<"Total occurances is: " << TotalOccurance <<endl;
    return 0;
}

//-------------------------------------------------------------------------------------------
---

// Program to find missing element and let range be 1-N Given i/p elements in array is N-1

#include <iostream>
#include<vector>
using namespace std;
int missingElement(vector<int>&v)
{
    int s = 0;
    int e = v.size() - 1;
    int mid = s + (e-s)/2;
    int ans = -1;

    while(s <= e)
    {
        if(v[mid] - mid == 1)
        {
            s = mid + 1;
        }
        else
        {
            ans = mid;
            e = mid - 1;
        }
        mid = s + (e-s)/2;
    }
    if(ans + 1 == 0)
    {
        return v.size() + 1;
    }
    return ans + 1;
}
int main()
{
    vector<int>v = {1,2,3,4,5,6,8,9,10};
    int missing = missingElement(v);

    cout << "Missing element is: " << missing << endl;
}

//------------------------------------------------------------------------------------------
---
```