Ques 1.Implement the Binary search algorithm regarded as a fast search algorithm with run-time complexity of O(log n) in comparison to the Linear Search.

Solution->#include <iostream>

```
using namespace std;

int binarySearch(int arr[], int size, int target) {
    int left = 0, right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target)
            return mid;

    if (arr[mid] < target)
            left = mid + 1;
        else
            right = mid - 1; // If target smaller, ignore right half
    }
```

```
    return -1; // Target not found
}

int main() {
    int arr[] = {2, 4, 10, 15, 23, 38, 56, 72, 91};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 23;

    int result = binarySearch(arr, size, target);

    if (result != -1)
        cout << "Element found at index " << result << endl;
    else
        cout << "Element not found in the array." << endl;

    return 0;
}
```

Output->Element found at index 4

Ques 2) Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. Code the Bubble sort with the following elements: 64 34 25 12 22 11 90

Solution->#include <iostream>

```
using namespace std;

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        // Last i elements are already in place
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
```

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

    cout << "Original array: ";
    printArray(arr, n);

    bubbleSort(arr, n);

    cout << "Sorted array: ";
    printArray(arr, n);

    return 0;
}
```

Output->Original array: 64 34 25 12 22 11 90

Sorted array: 11 12 22 25 34 64 90

Ques 3) Given an array of n-1 distinct integers in the range of 1 to n, find the missing number in it in a Sorted Array (a) Linear time (b) Using binary search.

Solution-> a)

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

int findMissingLinear(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        if (arr[i] != i + 1)
```

```
            return i + 1;
        }
    return n;
}

int main() {
    int arr[] = {1, 2, 3, 4, 6, 7, 8};
    int n = 8;

    cout << "Missing number (Linear): " << findMissingLinear(arr, n) <<
endl;
    return 0;
}
```

b)

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

int findMissingBinary(int arr[], int n) {
    int left = 0, right = n - 2;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == mid + 1)
```

```
            left = mid + 1;
        else
            right = mid - 1;
    }

    return left + 1;
}

int main() {
    int arr[] = {1, 2, 3, 4, 6, 7, 8};
    int n = 8;

    cout << "Missing number (Binary Search): " <<
findMissingBinary(arr, n) << endl;
    return 0;
}
```

Output->Missing number (Linear): 5

Missing number (Binary Search): 5