Order-Agnostic Binary Search
Steps of Order-Agnostic Binary Search

1. Determine the order of the array:
    - Compare the first and last elements of the array.
    - If arr[0] < arr[n-1], the array is sorted in ascending order.
    - Otherwise, it is sorted in descending order.
2. Perform Binary Search:
    - Modify the comparison conditions based on the array's order:
        - Ascending order: arr[mid] < target.
        - Descending order: arr[mid] > target.

Algorithm
1. Initialize `low = 0` and `high = n - 1`.
2. Determine the order of the array:
  - Ascending if `arr[low] < arr[high]`.
  - Descending otherwise.
3. Perform binary search:
  - Calculate `mid = low + (high - low) / 2`.
  - If `arr[mid] == target`, return `mid`.
  - Adjust `low` and `high` based on the order:
    - Ascending:
      - `low = mid + 1` if `arr[mid] < target`.
      - `high = mid - 1` if `arr[mid] > target`.
    - Descending:
      - `low = mid + 1` if `arr[mid] > target`.
      - `high = mid - 1` if `arr[mid] < target`.
4. If the element is not found, return `-1`.

Order agnostic binary search in c++
Sample Input
5
10 20 30 40 50
50
Your Output
Element found at index 4

Sample Input
5
50 40 30 20 10
50
Your Output
Element found at index 0

Order agnostic binary search in c++

```
#include <iostream>
#include <vector>
```

```
using namespace std;

int orderAgnosticBinarySearch(const vector<int>& arr, int target) {
    int low = 0, high = arr.size() - 1;

    // Determine if the array is ascending or descending
    bool isAscending = arr[low] < arr[high];

    while (low <= high) {
        int mid = low + (high - low) / 2;

        // Check if the target is found
        if (arr[mid] == target) {
            return mid;
        }

        if (isAscending) {
            // Ascending order logic
            if (arr[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        } else {
            // Descending order logic
            if (arr[mid] > target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
    }

    return -1; // Target not found
}

int main() {
    int n, target;
    cin >> n;

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

    cin >> target;

    int result = orderAgnosticBinarySearch(arr, target);
```

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

    return 0;
}
```

Order agnostic binary search in java

```
import java.util.Scanner;

class OrderAgnosticBinarySearch {

    public static int orderAgnosticBinarySearch(int[] arr, int target) {
        int low = 0, high = arr.length - 1;

        // Determine if the array is ascending or descending
        boolean isAscending = arr[low] < arr[high];

        while (low <= high) {
            int mid = low + (high - low) / 2;

            // Check if the target is found
            if (arr[mid] == target) {
                return mid;
            }

            if (isAscending) {
                // Ascending order logic
                if (arr[mid] < target) {
                    low = mid + 1;
                } else {
                    high = mid - 1;
                }
            } else {
                // Descending order logic
                if (arr[mid] > target) {
                    low = mid + 1;
                } else {
                    high = mid - 1;
                }
            }
        }

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

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[] arr = new int[n];

        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        int target = sc.nextInt();

        int result = orderAgnosticBinarySearch(arr, target);

        if (result != -1) {
            System.out.println("Element found at index " + result);
        } else {
            System.out.println("Element not found in the array.");
        }

        sc.close();
    }
}
```