

**Q1. Implementation of linear search**

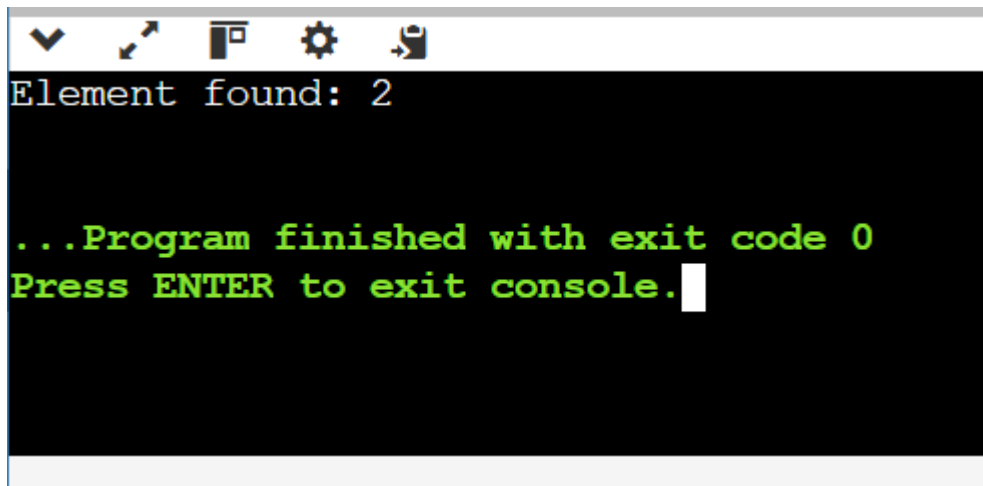
```
#include <iostream>

using namespace std;

int linearSearch(int arr[], int size, int t) {
    for (int i = 0; i < size; ++i) {
        if (arr[i] == t) {
            return i;
        }
    }
    return -1;
}

int main() {
    int arr[] = {1, 5, 14, 18, 25};
    int size = sizeof(arr) / sizeof(arr[0]);
    int t = 14;
    int result = linearSearch(arr, size, t);
    if (result != -1) {
        cout << "Element found: " << result << endl;
    } else {
        cout << "Element not found: " << endl;
    }
    return 0;
}
```

Output



```
Element found: 2

...Program finished with exit code 0
Press ENTER to exit console.
```

## Q2. Implementation of binary search to find index value

```
#include <iostream>

using namespace std;

int binarySearch(int arr[], int size, int t) {
    int l = 0, r = size - 1;
    while (l <= r) {
        int mid = l + (r - l) / 2;
        if (arr[mid] == t) {
            return mid;
        } else if (arr[mid] < t) {
            l = mid + 1;
        } else {
            r = mid - 1;
        }
    }
    return -1;
}

int main() {
```

```

int arr[] = {1, 4, 14, 25, 4, 18};

int size = sizeof(arr) / sizeof(arr[0]);

int t = 18 ;

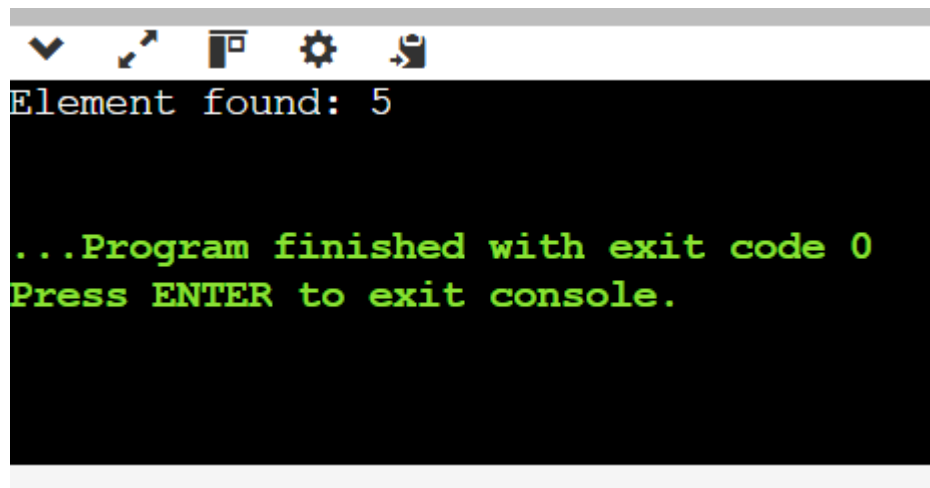
int result = binarySearch(arr, size, t);

if (result != -1) {
    cout << "Element found: " << result << endl;
} else {
    cout << "Element not found:" << endl;
}

return 0;
}

```

Output



```

Element found: 5

...Program finished with exit code 0
Press ENTER to exit console.

```

**Q3. Binary search to find first occurrence of target value in sorted array**

```

#include <iostream>

using namespace std;

int firstOccurrenceBinarySearch(int arr[], int size, int t) {
    int l = 0, r = size - 1;

```

```

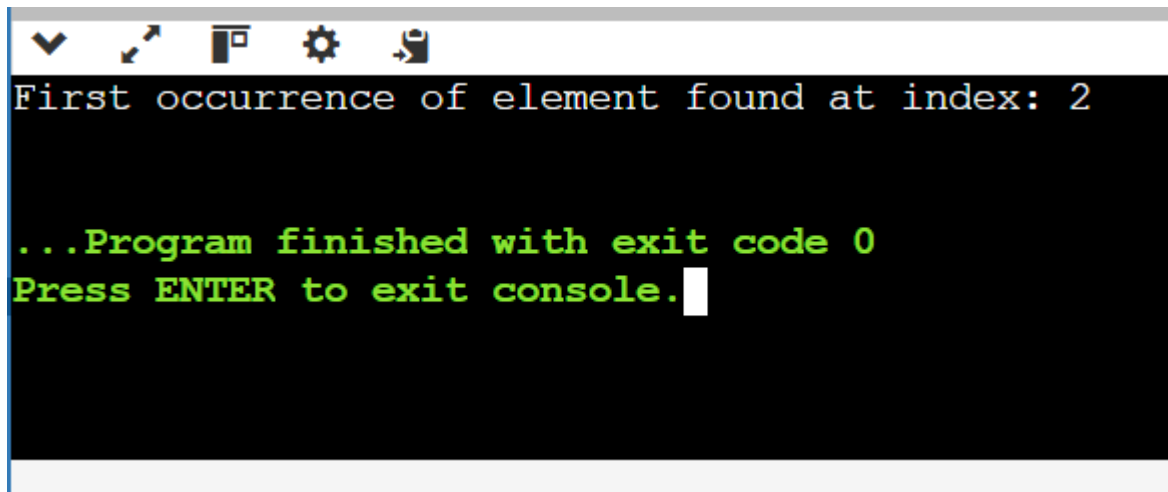
int result = -1;
while (l <= r) {
    int mid = l + (r - l) / 2;
    if (arr[mid] == t) {
        result = mid;
        r = mid - 1;
    } else if (arr[mid] < t) {
        l = mid + 1;
    } else {
        r = mid - 1;
    }
}

return result;
}

int main() {
    int arr[] = {5, 10, 15, 20, 25, 30};
    int size = sizeof(arr) / sizeof(arr[0]);
    int t = 12;
    int result = firstOccurrenceBinarySearch(arr, size, t);
    if (result != -1) {
        cout << "First occurrence of element found at index: " << result << endl; // Output: 2
    } else {
        cout << "Element not found:" << endl;
    }
    return 0;
}

```

## Output

A screenshot of a console window with a dark background. The text is displayed in a monospaced font. The first line is "First occurrence of element found at index: 2" in a light blue color. The second line is "...Program finished with exit code 0" in a light green color. The third line is "Press ENTER to exit console." in the same light green color, followed by a white cursor block.

```
First occurrence of element found at index: 2

...Program finished with exit code 0
Press ENTER to exit console.
```

Q4. appears only once in sorted array (bs)

```
#include <iostream>

using namespace std;

int singleNonDuplicate(int arr[], int size) {

    int left = 0, right = size - 1;

    while (left < right) {

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

        if (mid % 2 == 1) {

            mid--;

        }

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

            left = mid + 2;

        } else {

            right = mid;

        }

    }

    return arr[left];

}
```

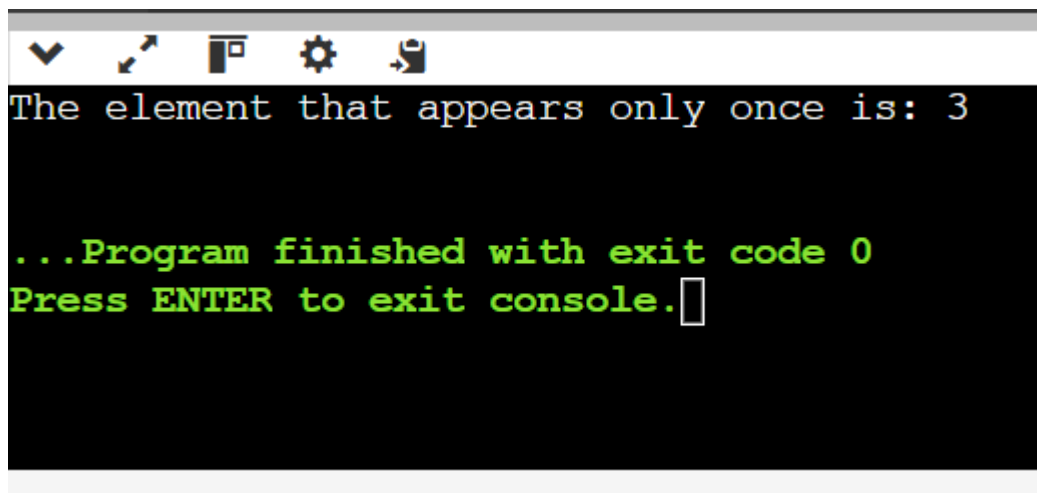
```

}

int main() {
    int arr[] = {1,1,2,2,3,4,4,5,5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int result = singleNonDuplicate(arr, size);
    cout << "The element that appears only once is: " << result << endl;
    return 0;
}

```

Output



The screenshot shows a console window with a dark background. At the top, there is a toolbar with icons for a dropdown menu, a cursor, a window, a gear, and a document. Below the toolbar, the text "The element that appears only once is: 3" is displayed in a light blue font. Below this, the text "...Program finished with exit code 0" is displayed in a green font. At the bottom, the text "Press ENTER to exit console." is displayed in a green font, followed by a small white square icon.

Q5. given an array sorted in ascending order and an integer k return true if k is present in the array otherwise false

```

#include <iostream>

using namespace std;

int binarySearch(int arr[], int size, int k) {
    int left = 0, right = size - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == k) {

```

```

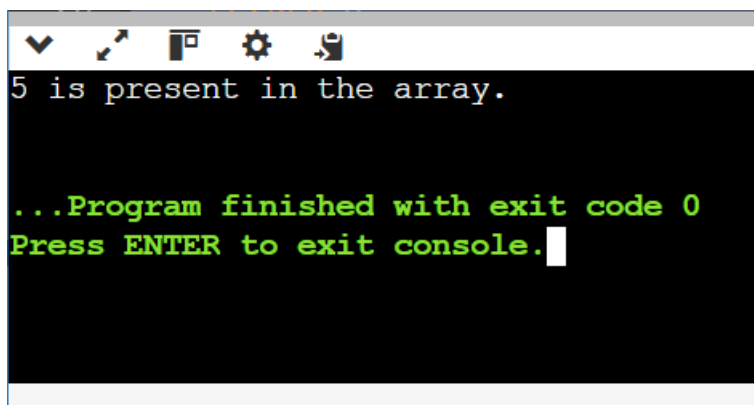
        return true;
    } else if (arr[mid] < k) {
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}

return false;
}

int main() {
    int arr[] = {1, 3, 5, 7, 9, 11, 13};
    int size = sizeof(arr) / sizeof(arr[0]);
    int k = 5;
    if (binarySearch(arr, size, k)) {
        cout << k << " is present in the array." << endl;
    } else {
        cout << k << " is not present in the array." << endl;
    }
    return 0;
}

```

Output



```

5 is present in the array.

...Program finished with exit code 0
Press ENTER to exit console.

```

## Q6. Bubble sort

```
#include <iostream>

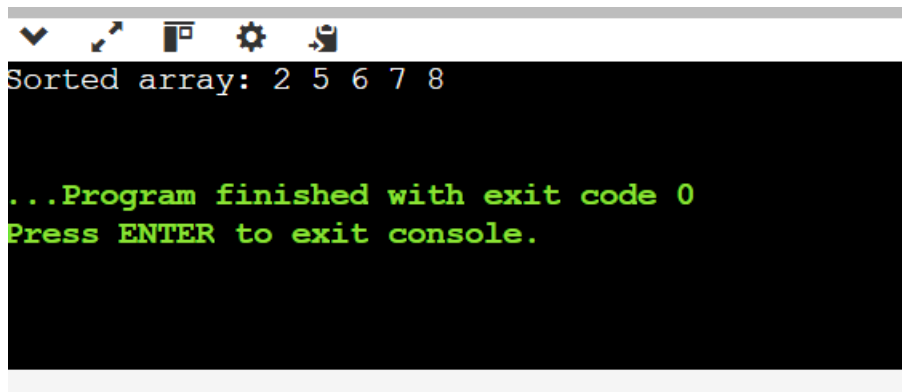
using namespace std;

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; ++i) {
        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;
            }
        }
    }
}

int main() {
    int arr[] = {8, 5, 7, 6, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    bubbleSort(arr, n);
    cout << "Sorted array: ";
    for (int i = 0; i < n; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Output



A screenshot of a terminal window with a dark background. The top bar shows standard window controls (minimize, maximize, close) and system icons. The terminal text is as follows:

```
Sorted array: 2 5 6 7 8

...Program finished with exit code 0
Press ENTER to exit console.
```

## Q7. Sum of binary tree nodes

```
#include <iostream>

using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int sumOfNodes(TreeNode* root) {
    if (root == nullptr) {
        return 0;
    }
    return root->val + sumOfNodes(root->left) + sumOfNodes(root->right);
}

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
```

```

    root->left->right = new TreeNode(5);
    root->right->right = new TreeNode(6);
    return root;
}

int main() {
    TreeNode* root = createExampleTree();

    int sum = sumOfNodes(root);

    cout << "Sum: " << sum << endl;

    return 0;
}

```

Output



```

Sum: 21

...Program finished with exit code 0
Press ENTER to exit console.

```

Q8. Find the tree is symmetric or not .Input – [1,2,2,3,4,4,3]

```

#include <iostream>

using namespace std;

struct TreeNode {
    int val;

    TreeNode* left;

    TreeNode* right;

    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
}

```

```

};

int isMirror(TreeNode* left, TreeNode* right) {
    if (left == nullptr && right == nullptr) {
        return true;
    }
    if (left == nullptr || right == nullptr) {
        return false;
    }
    return (left->val == right->val) && isMirror(left->left, right->right) && isMirror(left->right,
right->left);
}

int isSymmetric(TreeNode* root) {
    if (root == nullptr) {
        return true;
    }
    return isMirror(root->left, root->right);
}

TreeNode* createExampleTree() {
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(2);
    root->left->left = new TreeNode(3);
    root->left->right = new TreeNode(4);
    root->right->left = new TreeNode(4);
    root->right->right = new TreeNode(3);
    return root;
}

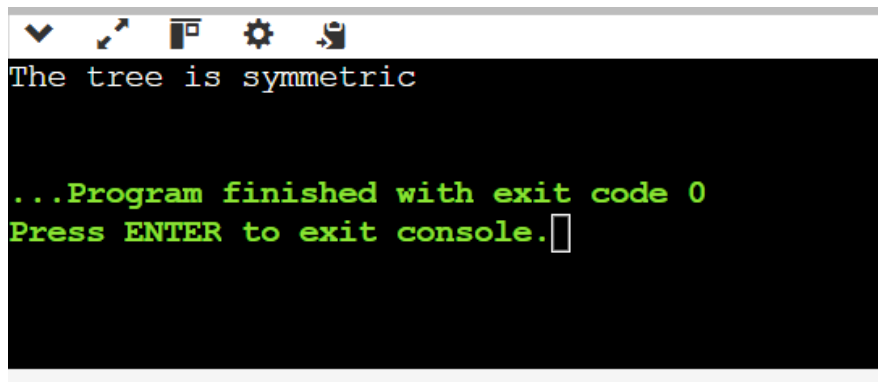
```

```

int main() {
    TreeNode* root = createExampleTree();
    bool symmetric = isSymmetric(root);
    cout << "The tree is " << (symmetric ? "symmetric" : "not symmetric") << endl;
    return 0;
}

```

Output



The screenshot shows a terminal window with a dark background. The text "The tree is symmetric" is displayed in white. Below it, in green, is the message "...Program finished with exit code 0" and "Press ENTER to exit console." followed by a cursor. The terminal window has a standard toolbar at the top with icons for window management and settings.

```

The tree is symmetric

...Program finished with exit code 0
Press ENTER to exit console.

```

## Q9. Squares of a Sorted Array

```

#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
vector<int> sortedSquares(vector<int>& nums) {
    int n = nums.size();
    vector<int> result(n);
    int left = 0, right = n - 1;
    int pos = n - 1;
    while (left <= right) {
        if (abs(nums[left]) > abs(nums[right])) {
            result[pos] = nums[left] * nums[left];

```

```

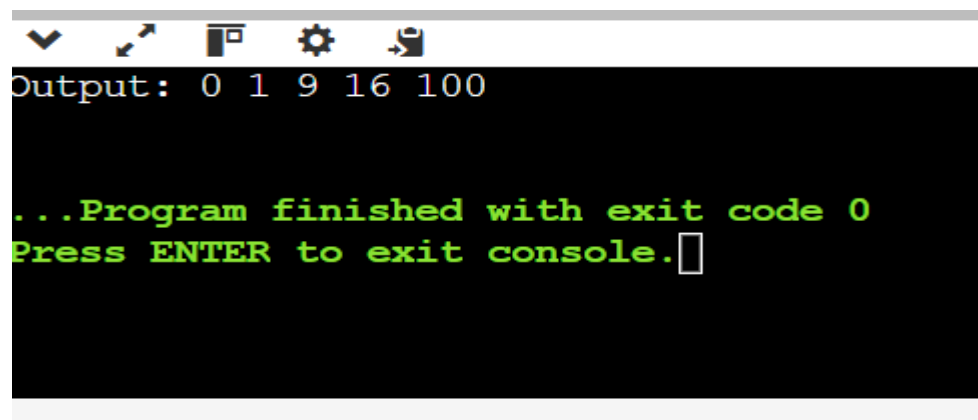
        left++;
    } else {
        result[pos] = nums[right] * nums[right];
        right--;
    }
    pos--;
}
return result;
}

int main() {
    vector<int> nums = {-4, -1, 0, 3, 10};
    vector<int> result = sortedSquares(nums);

    cout << "Output: ";
    for (int x : result) {
        cout << x << " ";
    }
    cout << endl;
    return 0;
}

```

Output



```

Output: 0 1 9 16 100

...Program finished with exit code 0
Press ENTER to exit console.

```

## Q10. Smallest positive missing number.

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int smallestMissingPositive(vector<int>& arr) {

    int n = arr.size();

    int j = 0;

    for (int i = 0; i < n; i++) {

        if (arr[i] <= 0) {

            swap(arr[i], arr[j]);

            j++;

        }

    }

    for (int i = j; i < n; i++) {

        int val = abs(arr[i]);

        if (val - 1 + j < n && arr[val - 1 + j] > 0) {

            arr[val - 1 + j] = -arr[val - 1 + j];

        }

    }

    for (int i = j; i < n; i++) {

        if (arr[i] > 0) {

            return i - j + 1;

        }

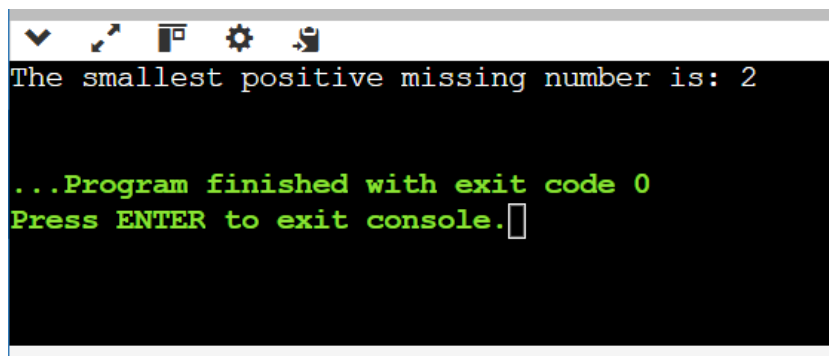
    }

    return n - j + 1;

}
```

```
int main() {  
    vector<int> arr = {3, 4, -1, 1};  
  
    int result = smallestMissingPositive(arr);  
  
    cout << "The smallest positive missing number is: " << result << endl;  
  
    return 0;  
}
```

Output

A screenshot of a console window with a black background and white text. The window has a title bar with standard icons. The output text is: "The smallest positive missing number is: 2" followed by a blank line, then "...Program finished with exit code 0" and "Press ENTER to exit console." with a cursor at the end.

```
The smallest positive missing number is: 2  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```