Name – Shreyansh Vishnoi
UID – 22BCS15373
Section – 620-B

DAY – 6

Searching a Number
Given an integer k and array arr. Your task is to return the position of the first occurrence of k in the given array and if element k is not present in the array then return -1.
Note: 1-based indexing is followed here.
Example1:
Input: k = 16 , arr = [9, 7, 16, 16, 4]
Output: 3
Explanation: The value 16 is found in the given array at positions 3 and 4, with position 3 being the first occurrence.
Example2:
Input: k=98 , arr = [1, 22, 57, 47, 34, 18, 66]
Output: -1

Example2:
Input: k=9 , arr = [1, 22, 57, 47, 34, 9, 66]
Output: 6

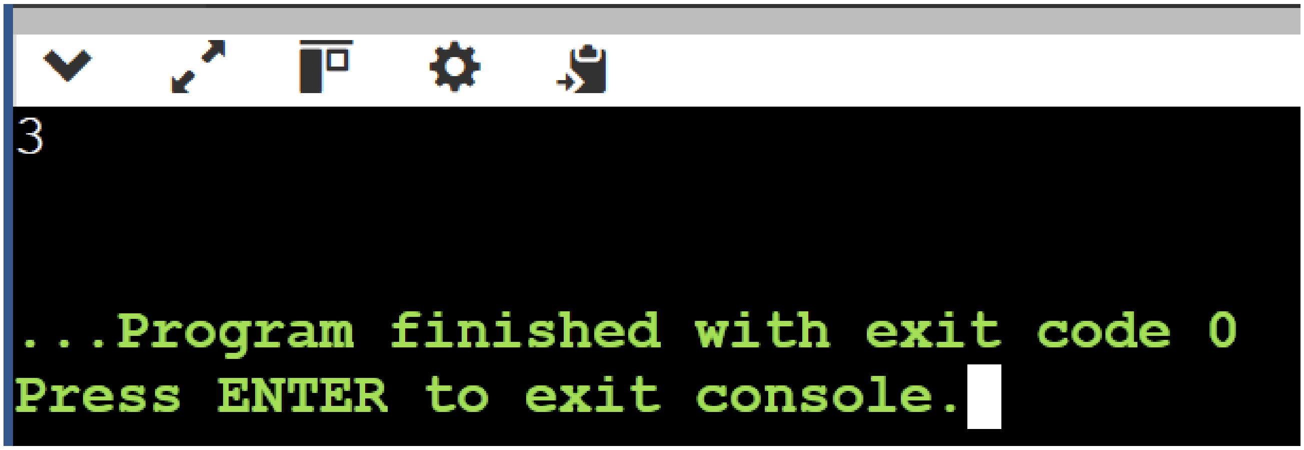
Explanation: k = 98 isn't found in the given array.
Expected Time Complexity: O(n)
Expected Auxiliary Space: O(1)
Constraints:

- **1 <= arr.size <= 10^6**
- **1 <= arr[i] <= 10^9**
- **1 <= k <= 10^6**

Reference:: https://www.geeksforgeeks.org/problems/searching-a-number0324/1

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

int searchNumber(int k, const vector<int>& arr) {
  // Iterate through the array to find the first occurrence of k
  for (int i = 0; i < arr.size(); ++i) {
    if (arr[i] == k) {
      return i + 1;  // Return 1-based index
    }
  }
  return -1;  // If not found
}
```

```
int main() {
    // Hardcoded values for k and the array
    int k = 16;
    vector<int> arr = {9, 7, 16, 16, 4};

    // Call the search function
    int result = searchNumber(k, arr);

    // Output the result
    cout << result << endl;  // Output: 3
    return 0;
}
```



```
3


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

2. Sorted array Search.

Given an array, arr[] sorted in ascending order and an integer k. Return true if k is present in the array, otherwise, false.

Example 1:

Input: arr[] = [1,2,3,4,6], k=6

Output: true

Explanation: Since, 6 is present in the array at index4 (0-based indexing), Output is true.

Example 2:

Input: arr[] = [1, 2, 4, 5, 6], k = 3

Output: false

Example 3:

Input: arr[] = [1, 2, 4, 5, 6], k = 6

Output: true

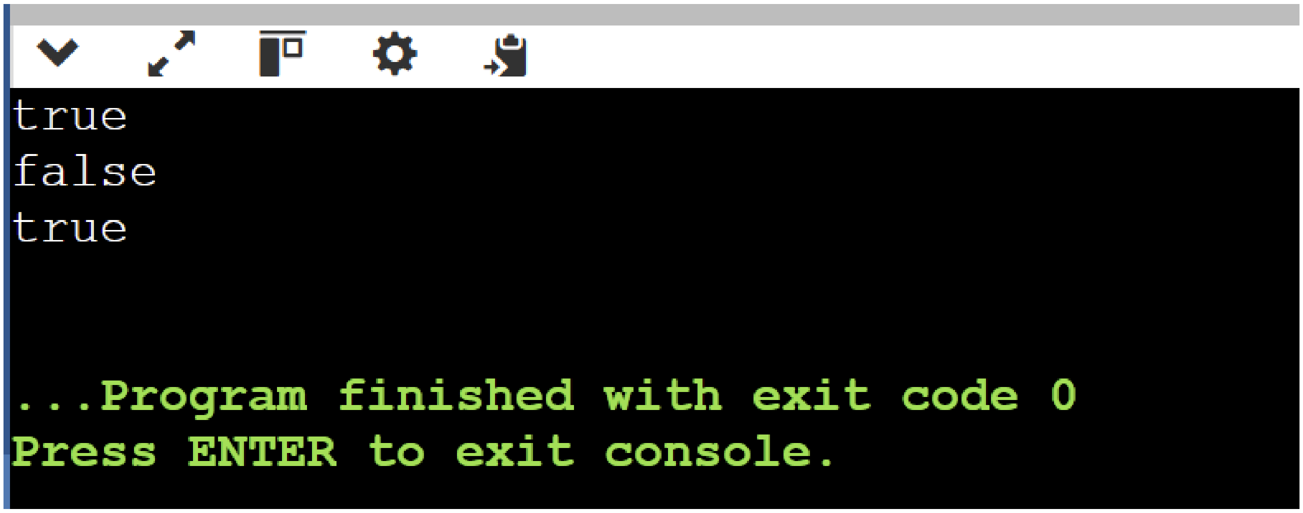
Exlpanation: Since, 3 is not present in the array, output is false.

Constraints:

- **1 <= arr.size() <= 10^6**
- **1 <= k <= 106**
- **1 <= arr[i] <= 10^6**

Reference: https://www.geeksforgeeks.org/problems/who-will-win-1587115621/1

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

// Function to perform binary search
bool binarySearch(const vector<int>& arr, int k) {
    int low = 0, high = arr.size() - 1;

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

        if (arr[mid] == k) {
            return true;  // k is found
        }
        else if (arr[mid] < k) {
            low = mid + 1;  // Search in the right half
        }
        else {
            high = mid - 1;  // Search in the left half
        }
    }

    return false;  // k is not found
}

int main() {
    // Hardcoded test cases
    vector<int> arr1 = {1, 2, 3, 4, 6};
    int k1 = 6;

    vector<int> arr2 = {1, 2, 4, 5, 6};
    int k2 = 3;

    vector<int> arr3 = {1, 2, 4, 5, 6};
    int k3 = 6;

    // Checking for presence of k in the array
    cout << (binarySearch(arr1, k1) ? "true" : "false") << endl;  // Output: true
    cout << (binarySearch(arr2, k2) ? "true" : "false") << endl;  // Output: false
    cout << (binarySearch(arr3, k3) ? "true" : "false") << endl;  // Output: true

    return 0;
}
```



```
true
false
true


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

3. Find Target Indices After Sorting Array.

You are given a 0-indexed integer array nums and a target element target.

A target index is an index i such that nums[i] == target.

Return a list of the target indices of nums after sorting nums in non-decreasing order. If there are no target indices, return an empty list. The returned list must be sorted in increasing order.

Example 1:

Input: nums = [1,2,5,2,3], target = 2

Output: [1,2]

Explanation: After sorting, nums is [1,2,2,3,5].

The indices where nums[i] == 2 are 1 and 2.

Example 2:

Input: nums = [1,2,5,2,3], target = 3

Output: [3]

Explanation: After sorting, nums is [1,2,2,3,5].

The index where nums[i] == 3 is 3.

Example 3:

Input: nums = [1,2,5,2,3], target = 5

Output: [4]

Explanation: After sorting, nums is [1,2,2,3,5].

The index where nums[i] == 5 is 4.

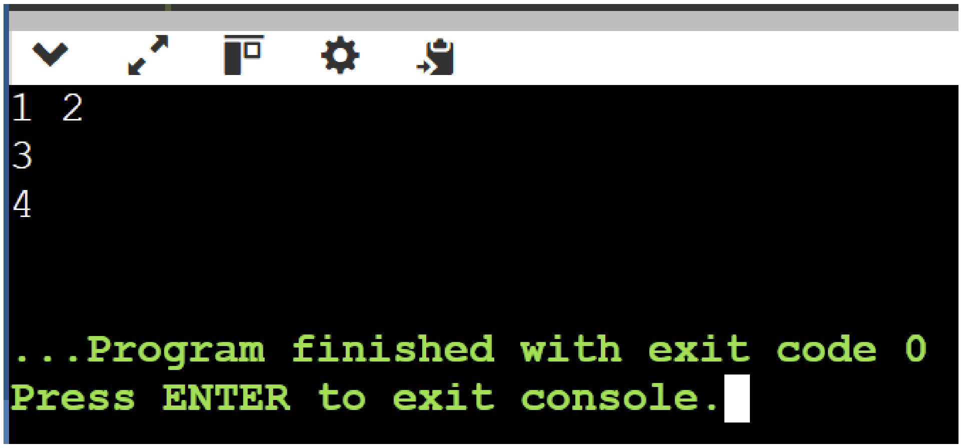
Constraints:

1 <= nums.length <= 100

1 <= nums[i], target <= 100

Reference:- https://leetcode.com/problems/find-target-indices-after-sorting-array/description/

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

vector<int> targetIndices(vector<int>& nums, int target) {
    // Sort the array
    sort(nums.begin(), nums.end());

    // Vector to store the target indices
    vector<int> indices;

    // Find all the indices where nums[i] == target
    for (int i = 0; i < nums.size(); ++i) {
        if (nums[i] == target) {
            indices.push_back(i);
        }
    }

    return indices;
}

int main() {
    // Test cases
    vector<int> nums1 = {1, 2, 5, 2, 3};
    int target1 = 2;
    vector<int> result1 = targetIndices(nums1, target1);
    for (int index : result1) {
        cout << index << " ";
    }
    cout << endl;  // Output: 1 2

    vector<int> nums2 = {1, 2, 5, 2, 3};
    int target2 = 3;
    vector<int> result2 = targetIndices(nums2, target2);
    for (int index : result2) {
        cout << index << " ";
    }
    cout << endl;  // Output: 3
    vector<int> nums3 = {1, 2, 5, 2, 3};
    int target3 = 5;
    vector<int> result3 = targetIndices(nums3, target3);
    for (int index : result3) {
        cout << index << " ";
    }
    cout << endl;  // Output: 4

    return 0;
}
```



```
1 2
3
4


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

4Search Insert Position.

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [1,3,5,6], target = 5

Output: 2

Example 2:

Input: nums = [1,3,5,6], target = 2

Output: 1

Example 3:

Input: nums = [1,3,5,6], target = 7

Output: 4

Constraints:

1 <= nums.length <= 10^4

-104 <= nums[i] <= 10^4

nums contains distinct values sorted in ascending order.

-104 <= target <= 10^4

Constraints:

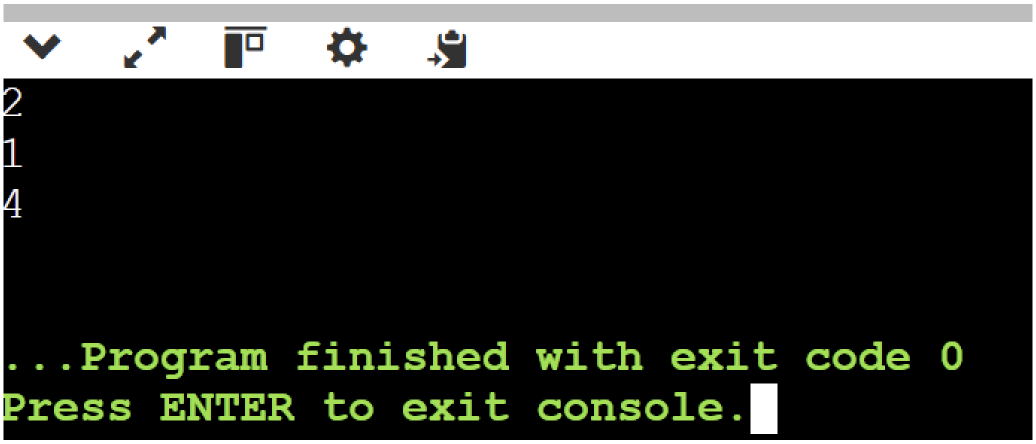
n == seats.length == students.length

1 <= n <= 100

1 <= seats[i], students[j] <= 100

Reference: https://leetcode.com/problems/search-insert-position/description/

```
#include <iostream>

#include <vector>

using namespace std;



int searchInsertPosition(vector<int>& nums, int target) {

    int low = 0, high = nums.size() - 1;



    // Perform binary search

    while (low <= high) {

        int mid = low + (high - low) / 2;
        if (nums[mid] == target) {
            return mid;  // If target is found, return its index
        } else if (nums[mid] < target) {
            low = mid + 1;  // Narrow search to the right half
        } else {
            high = mid - 1;  // Narrow search to the left half
        }
    }

    // If target is not found, return the position where it
should be inserted
    return low;  // After exiting the loop, 'low' will be the
insert position
}

int main() {
    // Test cases
    vector<int> nums1 = {1, 3, 5, 6};
    int target1 = 5;
    cout << searchInsertPosition(nums1, target1) << endl;  //
Output: 2

    vector<int> nums2 = {1, 3, 5, 6};
    int target2 = 2;
    cout << searchInsertPosition(nums2, target2) << endl;  //
Output: 1


    vector<int> nums3 = {1, 3, 5, 6};

    int target3 = 7;

    cout << searchInsertPosition(nums3, target3) << endl;  //
Output: 4


    return 0;

}
```



```
2
1
4


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

5. Relative Sort Array.

Given two arrays arr1 and arr2, the elements of arr2 are distinct, and all elements in arr2 are also in arr1.

Sort the elements of arr1 such that the relative ordering of items in arr1 are the same as in arr2. Elements that do not appear in arr2 should be placed at the end of arr1 in ascending order.

Example 1:

Input: arr1 = [2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6]

Output: [2,2,2,1,4,3,3,9,6,7,19]

Example 2:

Input: arr1 = [28,6,22,8,44,17], arr2 = [22,28,8,6]

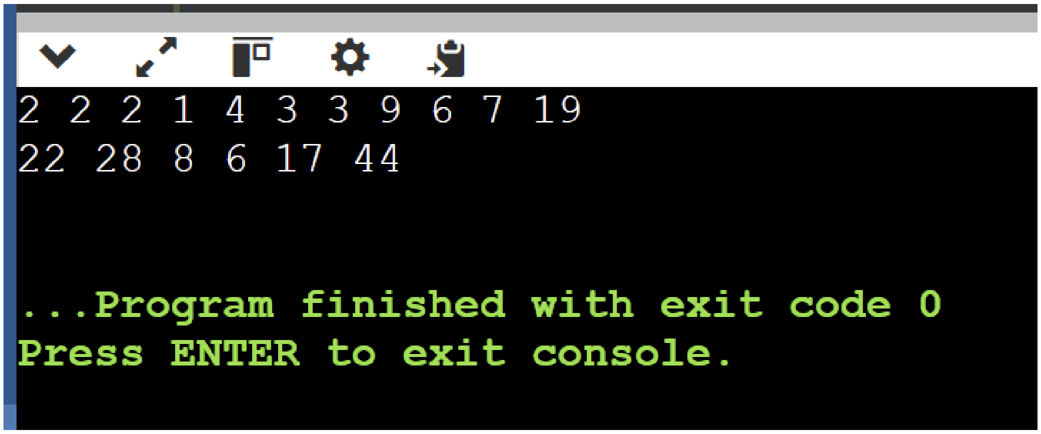
Output: [22,28,8,6,17,44]

Constraints:-

- **1 <= arr1.length, arr2.length <= 1000**
- **0 <= arr1[i], arr2[i] <= 1000**
- **All the elements of arr2 are distinct.**
- **Each arr2[i] is in arr1.**

Reference:-https://leetcode.com/problems/relative-sort-array/description/

```
#include <iostream>
#include <vector>
#include <unordered_map>
#include <algorithm>
using namespace std;

vector<int> relativeSortArray(vector<int>& arr1, vector<int>& arr2) {
    // Step 1: Create a frequency map for arr1
    unordered_map<int, int> freq;
    for (int num : arr1) {
        freq[num]++;
    }

    // Step 2: Create the result array
    vector<int> result;

    // Step 3: Add elements of arr2 to result based on frequency
    for (int num : arr2) {
        while (freq[num] > 0) {
            result.push_back(num);
            freq[num]--;
        }
    }

    // Step 4: Add the remaining elements that are not in arr2
    vector<int> remaining;
    for (auto& pair : freq) {
        while (pair.second > 0) {
            remaining.push_back(pair.first);
            pair.second--;
        }
    }

    // Step 5: Sort the remaining elements and add them to the result
    sort(remaining.begin(), remaining.end());
    result.insert(result.end(), remaining.begin(), remaining.end());

    return result;
}

int main() {
    // Test cases
    vector<int> arr1 = {2, 3, 1, 3, 2, 4, 6, 7, 9, 2, 19};
    vector<int> arr2 = {2, 1, 4, 3, 9, 6};
    vector<int> result = relativeSortArray(arr1, arr2);

    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;  // Output: [2, 2, 2, 1, 4, 3, 3, 9, 6, 7, 19]

    vector<int> arr3 = {28, 6, 22, 8, 44, 17};
    vector<int> arr4 = {22, 28, 8, 6};
    result = relativeSortArray(arr3, arr4);

    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;  // Output: [22, 28, 8, 6, 17, 44]

    return 0;
}}
```



```
2 2 2 1 4 3 3 9 6 7 19
22 28 8 6 17 44


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

6. We are given an array asteroids of integers representing asteroids in a row.

For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.

Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

Example 1:

Input: asteroids = [5,10,-5]

Output: [5,10]

Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.

Example 2:

Input: asteroids = [8,-8]

Output: []

Explanation: The 8 and -8 collide exploding each other.

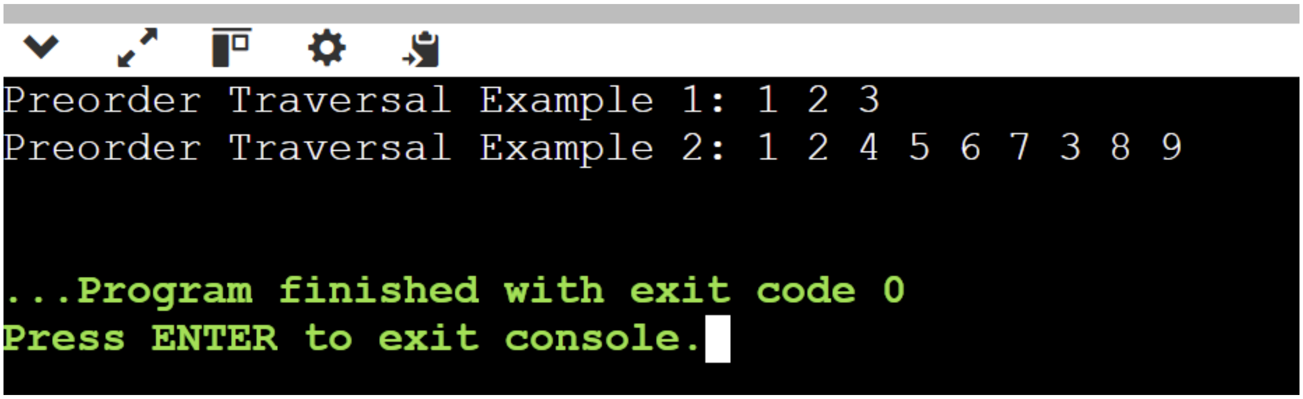
Example 3:

Input: asteroids = [10,2,-5]

Output: [10]

```
#include <iostream>

#include <vector>

#include <stack>
```

```
    }

using namespace std;

vector<int> asteroidCollision(vector<int>& asteroids) {
    stack<int> st; // Stack to simulate asteroid collisions

    for (int asteroid : asteroids) {
        bool destroyed = false;

        while (!st.empty() && asteroid < 0 && st.top() > 0) {
            if (abs(asteroid) == st.top()) {
                st.pop(); // Both asteroids explode
                destroyed = true;
                break;
            } else if (abs(asteroid) > st.top()) {
                st.pop(); // Top asteroid is smaller and explodes
            } else {
                destroyed = true; // Current asteroid is smaller and explodes
                break;
            }
        }

        if (!destroyed) st.push(asteroid); // Push the current asteroid if not destroyed
    }
    vector<int> result(st.size());
    for (int i = st.size() - 1; i >= 0; i--) {
        result[i] = st.top(); // Transfer stack to result in reverse order
        st.pop();
    }

    return result;
}

int main() {
    vector<int> asteroids1 = {5, 10, -5};
    vector<int> asteroids2 = {8, -8};
    vector<int> asteroids3 = {10, 2, -5};

    vector<int> result1 = asteroidCollision(asteroids1);
    vector<int> result2 = asteroidCollision(asteroids2);
    vector<int> result3 = asteroidCollision(asteroids3);

    for (int x : result1) cout << x << " "; // Output: 5 10
    cout << endl;
    for (int x : result2) cout << x << " "; // Output: (empty)
    cout << endl;
    for (int x : result3) cout << x << " "; // Output: 10

    cout << endl;



    return 0;

}
```



```
Preorder Traversal Example 1: 1 2 3
Preorder Traversal Example 2: 1 2 4 5 6 7 3 8 9


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

8 Binary Tree - Sum of All Nodes
Given the root of a binary tree, you need to find the sum of all the node values in the binary tree.

Example 1:
Input: root = [1, 2, 3, 4, 5, null, 6]

Output: 21

Explanation: The sum of all nodes is 1 + 2 + 3 + 4 + 5 + 6 = 21.

Example 2:
Input: root = [5, 2, 6, 1, 3, 4, 7]

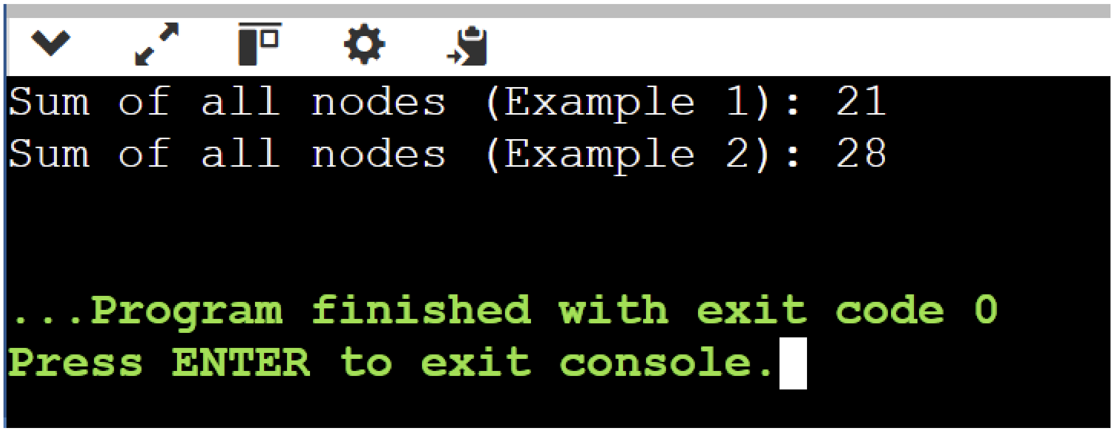
Output: 28

Explanation: The sum of all nodes is 5 + 2 + 6 + 1 + 3 + 4 + 7 = 28.

Reference: http://leetcode.com/problems/sum-of-left-leaves/

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

// Definition for a binary tree node.
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;

    TreeNode() : val(0), left(nullptr), right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
    TreeNode(int x, TreeNode* left, TreeNode* right) : val(x), left(left), right(right) {}
};

class Solution {
public:
    int sumOfNodes(TreeNode* root) {
        if (!root) return 0; // Base case: if the tree is empty, sum is 0
        return root->val + sumOfNodes(root->left) + sumOfNodes(root->right);
    }
};

int main() {
    // Example 1: [1, 2, 3, 4, 5, null, 6]
    TreeNode* root1 = new TreeNode(1);
    root1->left = new TreeNode(2);
```

```
root1->right = new TreeNode(3);
root1->left->left = new TreeNode(4);
root1->left->right = new TreeNode(5);
root1->right->right = new TreeNode(6);

Solution solution;
cout << "Sum of all nodes (Example 1): " << solution.sumOfNodes(root1) << endl;
// Output: 21

// Example 2: [5, 2, 6, 1, 3, 4, 7]
TreeNode* root2 = new TreeNode(5);
root2->left = new TreeNode(2);
root2->right = new TreeNode(6);
root2->left->left = new TreeNode(1);
root2->left->right = new TreeNode(3);
root2->right->left = new TreeNode(4);
root2->right->right = new TreeNode(7);

cout << "Sum of all nodes (Example 2): " << solution.sumOfNodes(root2) << endl;
// Output: 28

return 0;
```



```
Sum of all nodes (Example 1): 21
Sum of all nodes (Example 2): 28


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

10. Same Tree

Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

Example 1:
Input: p = [1,2,3], q = [1,2,3]
Output: true

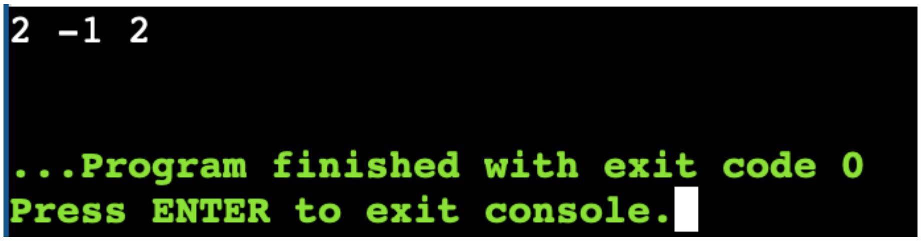
Example 2:
Input: p = [1,2], q = [1,null,2]
Output: false

Constraints:
The number of nodes in both trees is in the range [0, 100].
-104 <= Node.val <= 104

Reference: https://leetcode.com/problems/same-tree/description/?envType=study-plan-v2&envId=top-interview-150

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

// Definition for a binary tree node.
struct TreeNode {
   int val;
   TreeNode* left;
   TreeNode* right;

   TreeNode() : val(0), left(nullptr), right(nullptr) {}
   TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
   TreeNode(int x, TreeNode* left, TreeNode* right) : val(x), left(left), right(right) {}
};
```

```
class Solution {
public:
    bool isSameTree(TreeNode* p, TreeNode* q) {
        // If both trees are empty, they are the same
        if (!p && !q) return true;
        // If one is empty and the other is not, they are not the same
        if (!p || !q) return false;
        // If the values of the current nodes are different, they are not the same
        if (p->val != q->val) return false;

        // Recursively check the left and right subtrees
        return isSameTree(p->left, q->left) && isSameTree(p->right, q->right);
    }
};

int main() {
    // Example 1: p = [1, 2, 3], q = [1, 2, 3]
    TreeNode* p1 = new TreeNode(1);
    p1->left = new TreeNode(2);
    p1->right = new TreeNode(3);

    TreeNode* q1 = new TreeNode(1);
    q1->left = new TreeNode(2);
    q1->right = new TreeNode(3);

    Solution solution;
    cout << "Are trees p1 and q1 the same? " << (solution.isSameTree(p1, q1) ? "true" : "false") << endl;
    // Output: true

    // Example 2: p = [1, 2], q = [1, null, 2]
    TreeNode* p2 = new TreeNode(1);
    p2->left = new TreeNode(2);

    TreeNode* q2 = new TreeNode(1);
    q2->right = new TreeNode(2);

    cout << "Are trees p2 and q2 the same? " << (solution.isSameTree(p2, q2) ? "true" : "false") << endl;
    // Output: false

    return 0;
}
```



```
2 -1 2



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