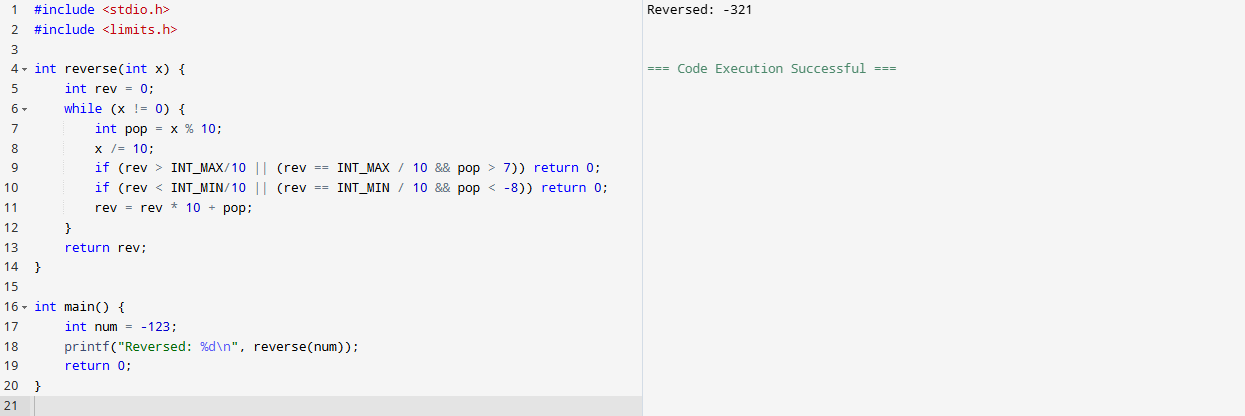
1. **Reversing a 32 bit signed integers.**

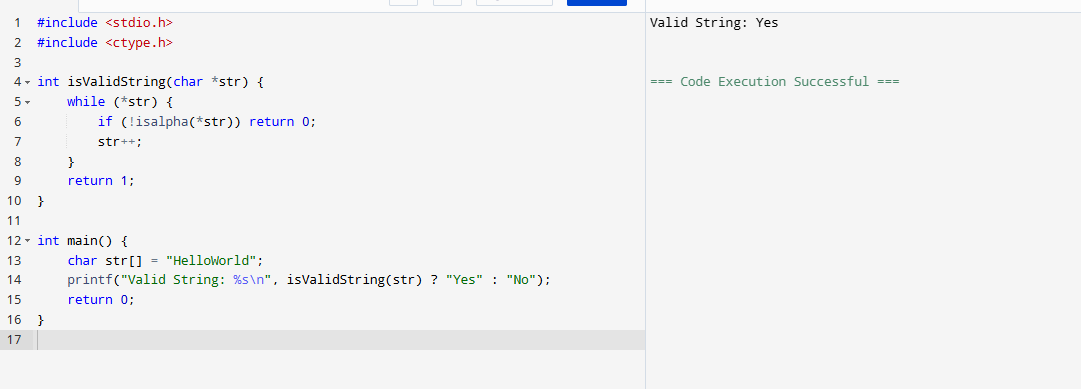
Aim: To reverse a 32-bit signed integer without overflow.



**Result:** The program was successfully executed and the reverse of the given 32-bit signed integer was calculated and displayed.

1. Check for a valid String

**Aim:** To check if a string is valid (contains only alphabets).



**Result:**  
The program was successfully executed and it verified that the input string contains only valid alphabetic characters.

1. Merging two Arrays

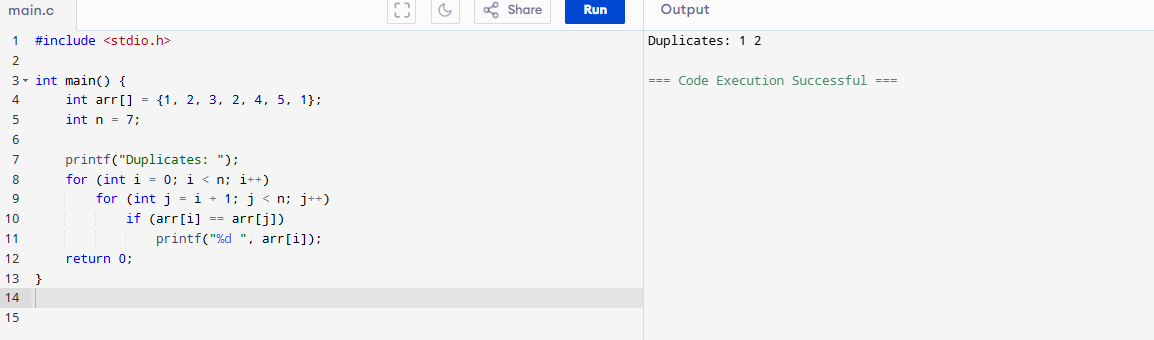
**Aim:** To merge two arrays into one.



**Result:**  
The program was successfully executed and the two arrays were merged into a single array and displayed.

1. Given an array finding duplication values

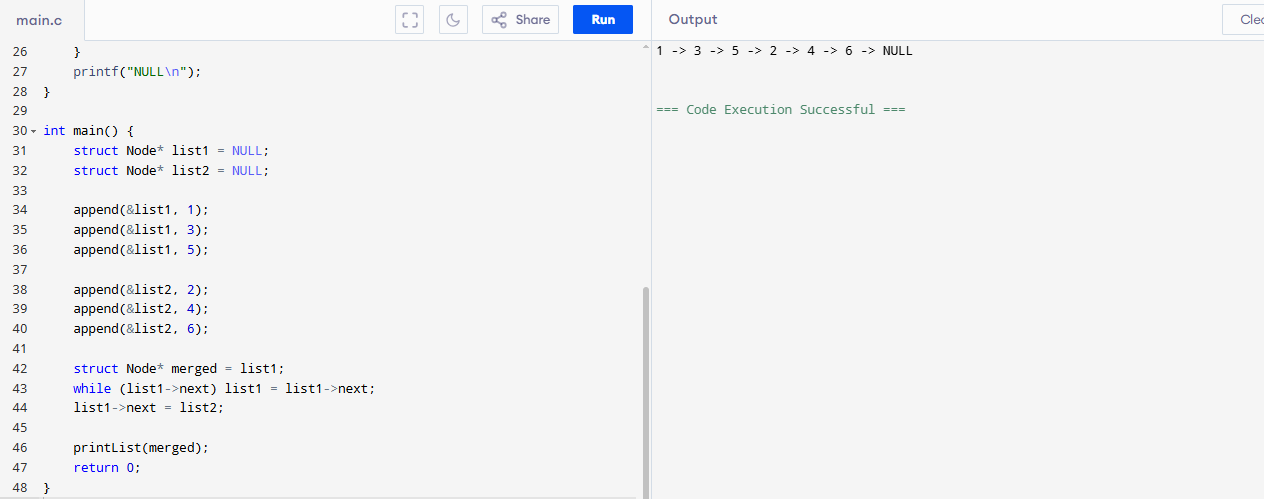
**Aim:** To find and print duplicate elements in an array.



**Result:**  
The program was successfully executed and the duplicate values in the given array were identified and displayed.

1. Merging of list

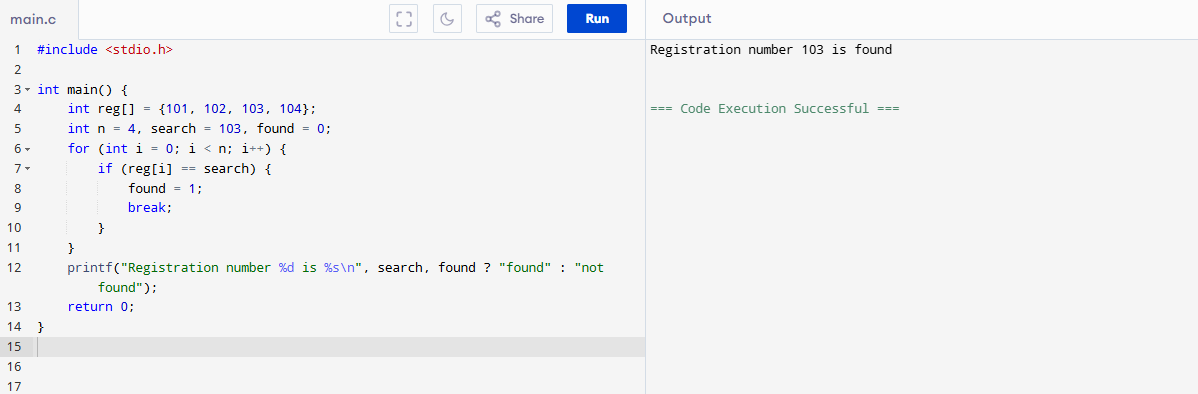
**Aim:** To merge two linked lists.



**Result:**  
The program was successfully executed and two linked lists were merged into one and the merged list was displayed.

1. Given array of reg nos need to search for particular reg no

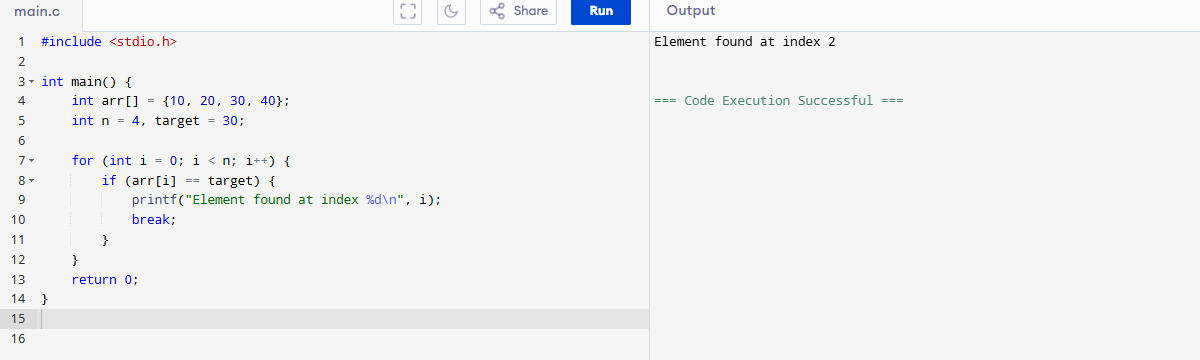
**Aim:** To search for a given registration number in an array.



**Result:**  
The program was successfully executed and it successfully searched and confirmed the presence of the specified registration number in the array.

1. Identify location of element in given array

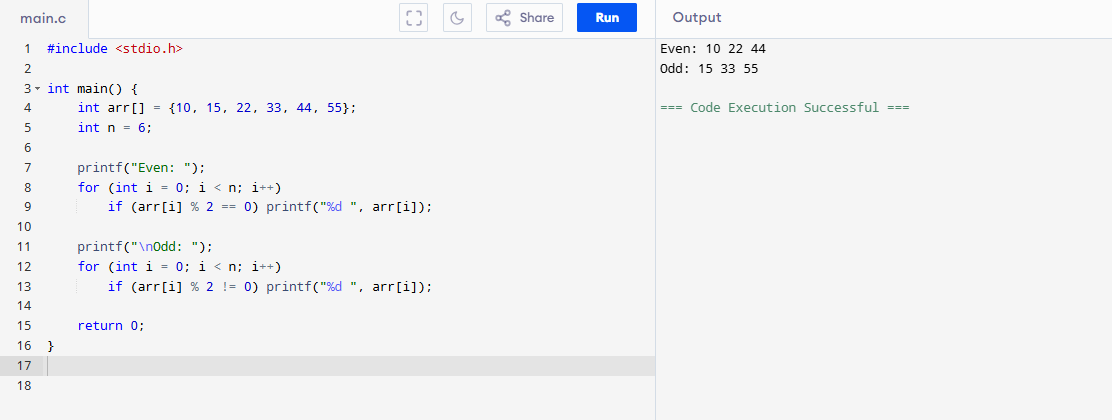
**Aim:**  
To find the index of a given element in an array.



**Result:**  
The program was successfully executed and the location (index) of the specified element in the array was identified and displayed.

1. Given array print odd and even values

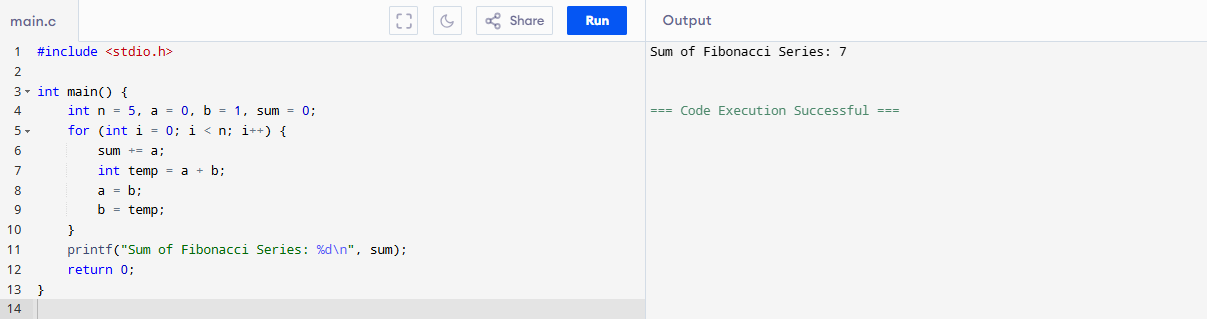
**Aim:**  
To separate and print odd and even numbers from an array.



**Result:**  
The program was successfully executed and it separated and displayed the odd and even elements from the given array.

9.sum of Fibonacci Series

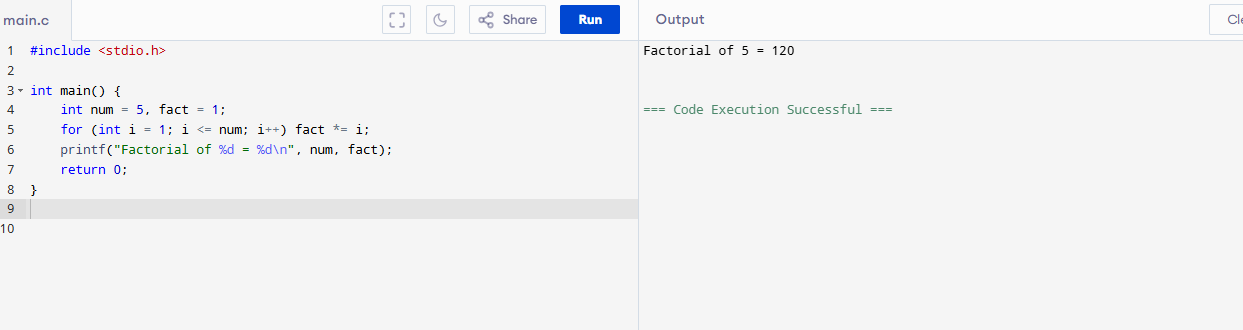
**Aim:**  
To find the sum of Fibonacci series up to n terms



**Result:**  
The program was successfully executed and the sum of the Fibonacci series up to the specified number of terms was calculated and displayed.

10.Finding factorial of a number

**Aim:**  
To calculate the factorial of a number using iteration.

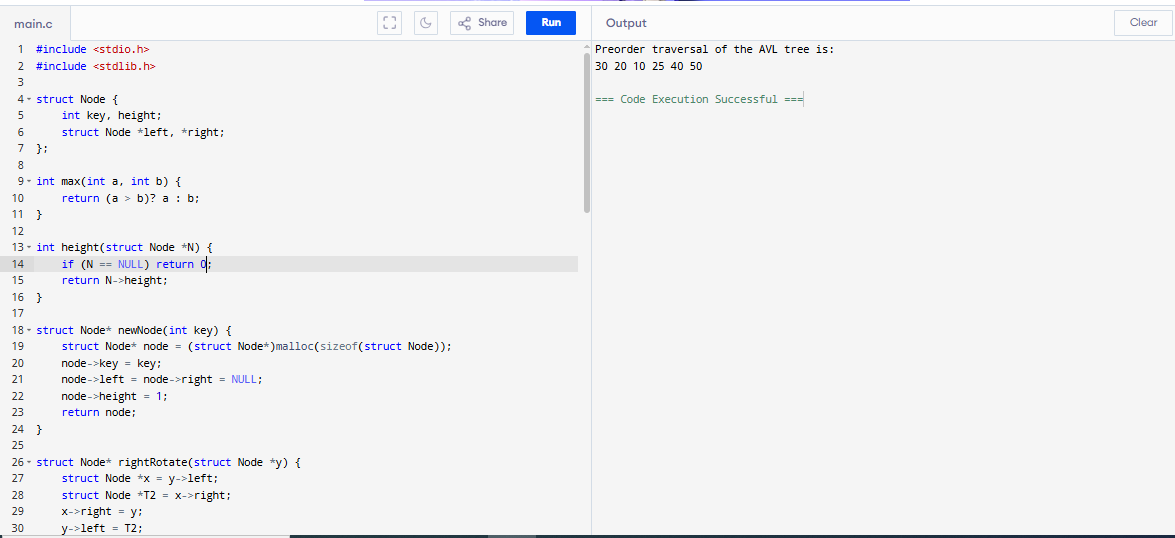


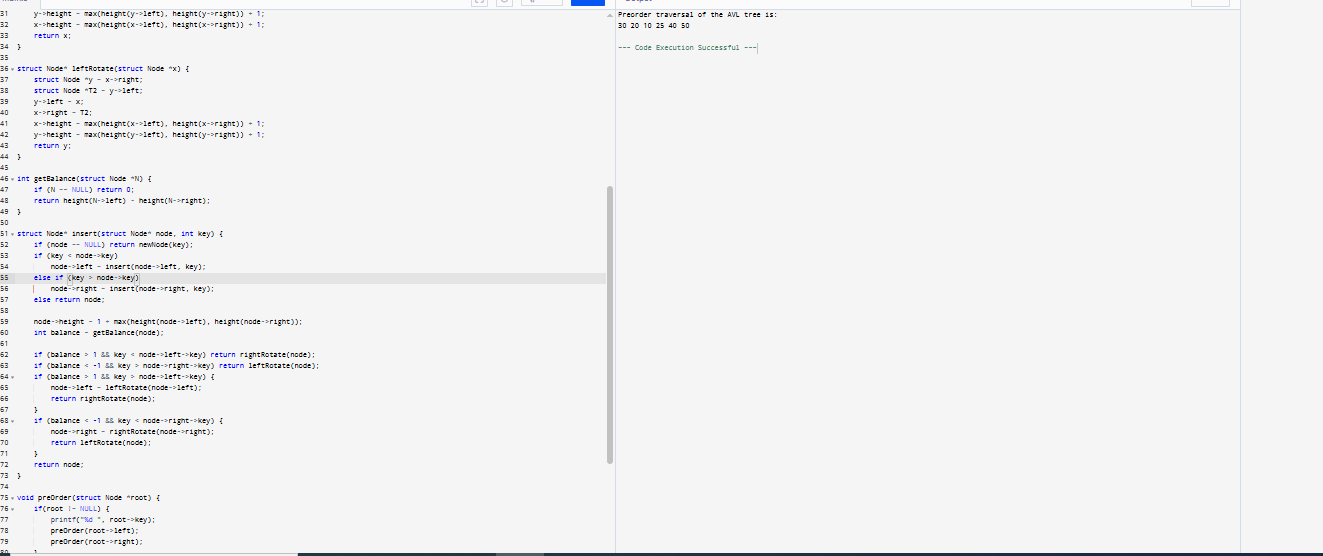
**Result:**  
The program was successfully executed and the factorial of the given number was calculated and displayed.

11. AVL tree

**Aim:**

To implement an AVL tree with insertion and balancing.





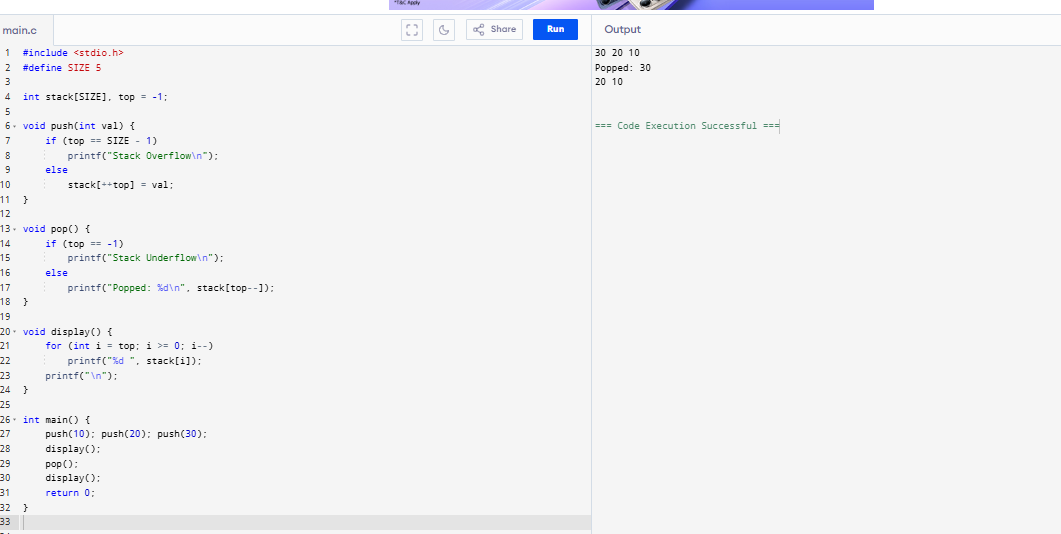
**Result :**

The program was successfully executed and the AVL tree was created with balanced insertions. The preorder traversal of the balanced tree was displayed.

12. Valid stack

**Aim:**

To implement a valid stack with push and pop operations.



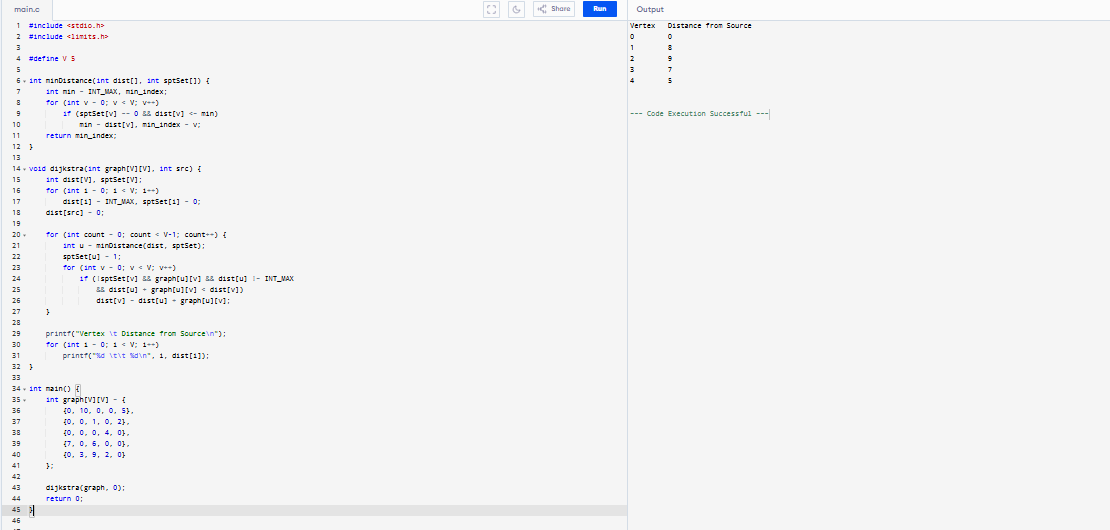
**Result in Words:**

The program was successfully executed and a valid stack was implemented with push and pop operations. The current stack content was displayed after each operation.

13. Graph - shortest path

**Aim:**

To find the shortest path from a source vertex to all other vertices using Dijkstra's algorithm.



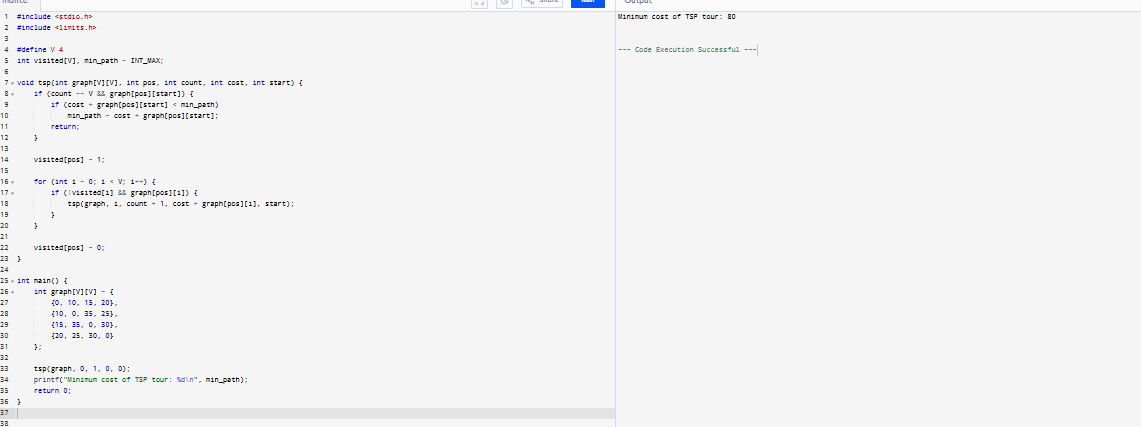
**Result :**

The program was successfully executed and the shortest path from the source vertex to all other vertices was calculated using Dijkstra’s algorithm.

14. Traveling Salesman Problem

**Aim:**

To solve the Traveling Salesman Problem using brute-force recursion for a small number of cities.



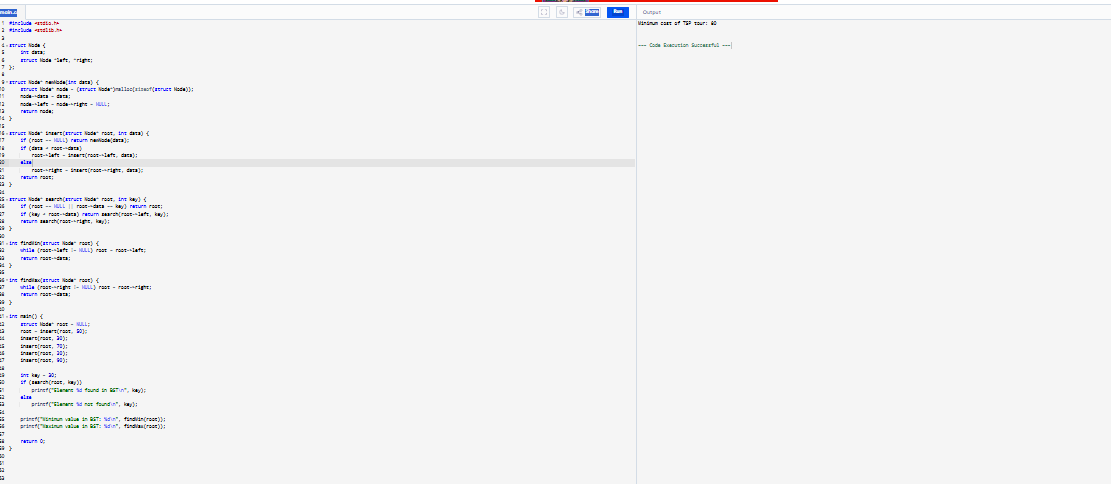
**Result:**

The program was successfully executed and the minimum cost path for the Traveling Salesman Problem was computed using brute-force recursion.

15. ! Binary search tree - search for a element, min element and Max element

**Aim:**

To implement a binary search tree (BST) and perform search, find minimum, and find maximum operations.



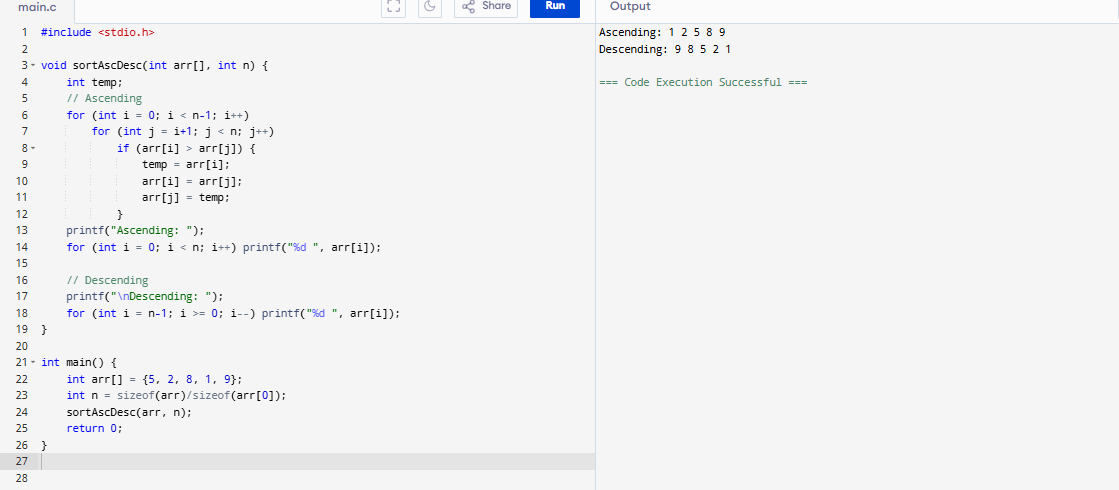
**Result :**

The program was successfully executed and the binary search tree was created. The specified element was searched, and the minimum and maximum values in the BST were displayed.

16. Array Sort – Ascending and Descending

**Aim:**

To sort an array in both ascending and descending order.



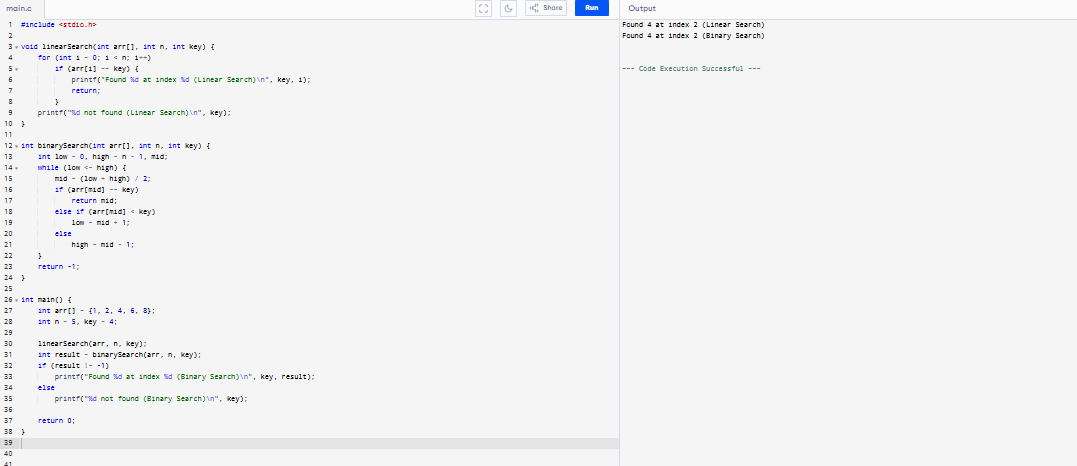
**Result :**

The program was successfully executed and the given array was sorted in both ascending and descending order.

17. Array Search – Linear and Binary Search

**Aim:**

To search for an element in an array using linear and binary search.



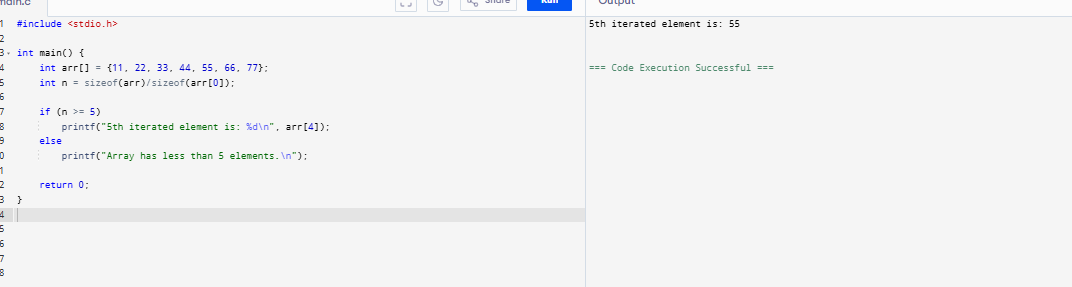
**Result :**

The program was successfully executed and the specified element was searched using both linear and binary search techniques.

18. Given Set of Array Elements – Display 5th Iterated Element

**Aim:**

To display the 5th element from a given array using array indexing



**Result :**

The program was successfully executed and the 5th iterated element of the array was displayed.

19. Given Unsorted Array – Display Missing Element

**Aim:**

To find and display the missing element in a sequence from an unsorted array.



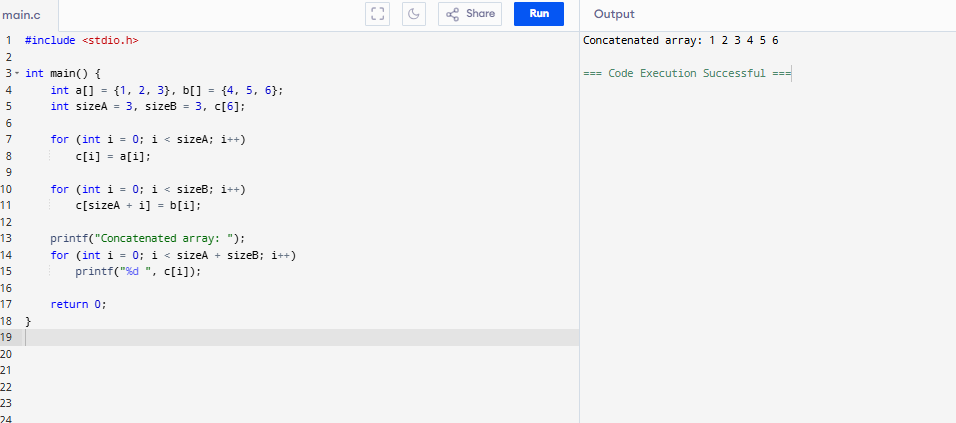
**Result :**

The program was successfully executed and the missing element in the given unsorted array was identified and displayed.

20.Array Concatenation

**Aim:**

To concatenate two arrays into a single array.



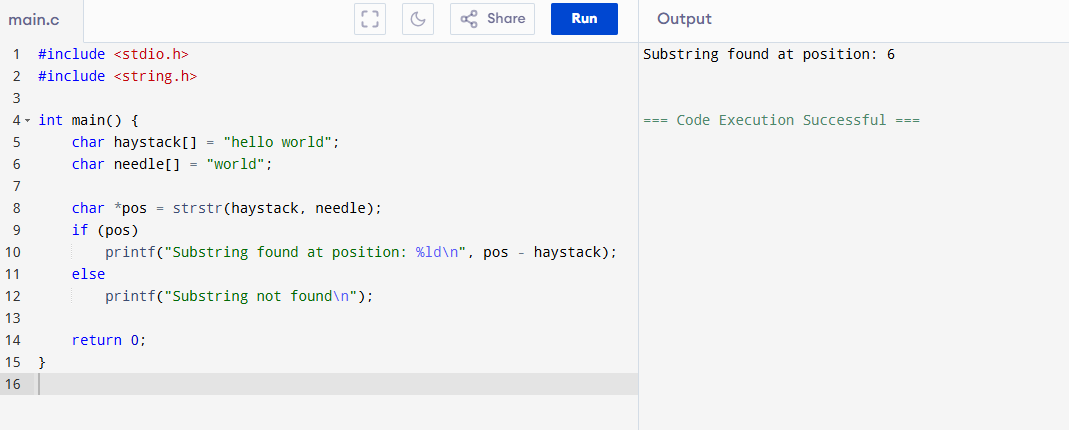
**Result :**

The program was successfully executed and the two arrays were concatenated into a single array and displayed.

21. Haystack – Substring Search

**Aim:**

To search for a substring (needle) in a main string (haystack).



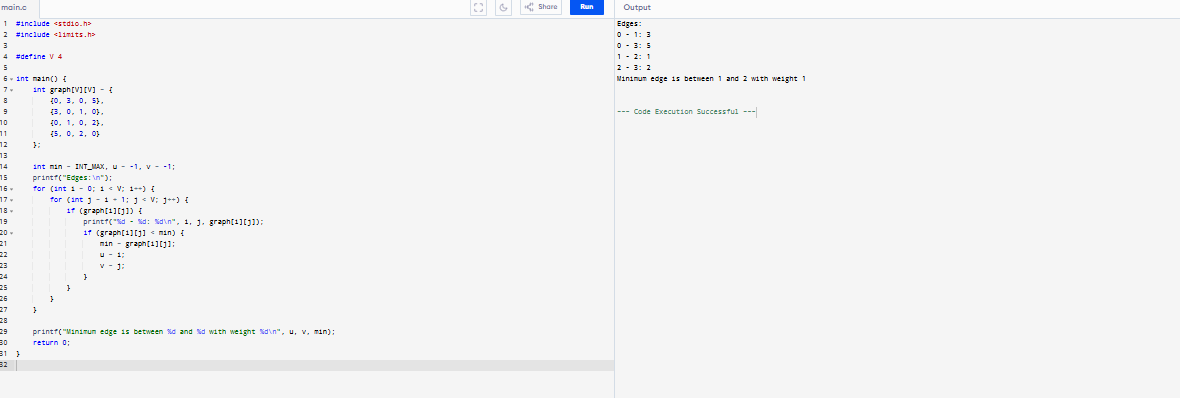
**Result :**

The program was successfully executed and the position of the substring (needle) in the main string (haystack) was found and displayed.

22. Given Graph – Convert to Array and Print Minimum Edges

**Aim:**

To convert a graph represented by an adjacency matrix to an edge list and print the minimum edge.



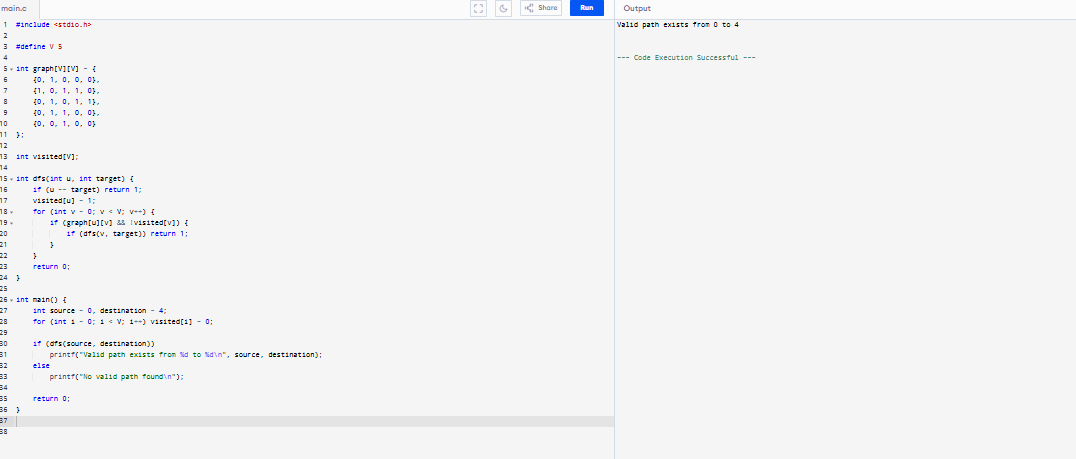
**Result :**

The program was successfully executed. The graph was converted to an edge list, and the edge with the minimum weight was identified and printed.

23. Given Graph – Print Valid Path

**Aim:**

To check and print if a path exists between two vertices in an unweighted graph.



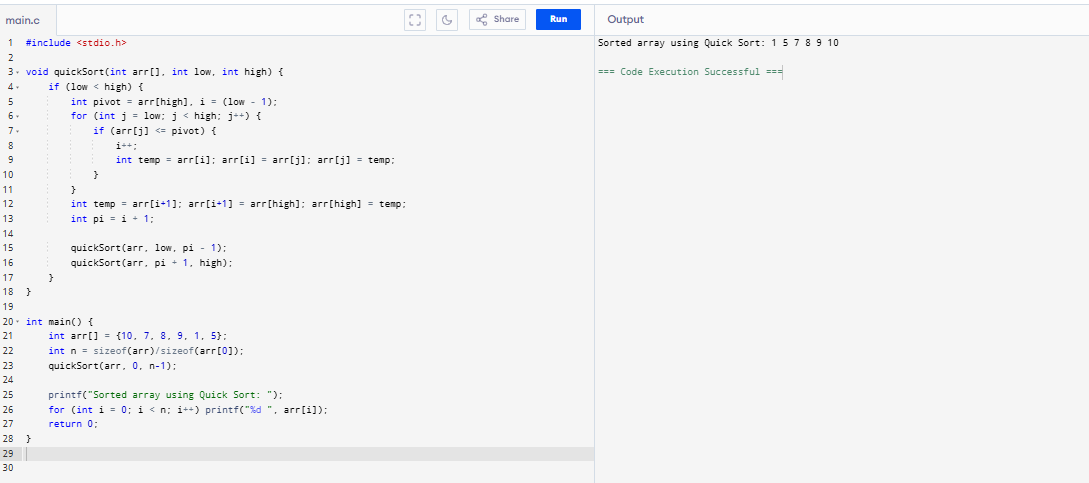
**Result :**

The program was successfully executed. It checked the connectivity between two nodes and confirmed that a valid path exists using Depth-First Search (DFS).

24. Heap, Merge, Insertion, and Quick Sort (Example: Quick Sort)

**Aim:**

To implement and demonstrate sorting using Quick Sort.



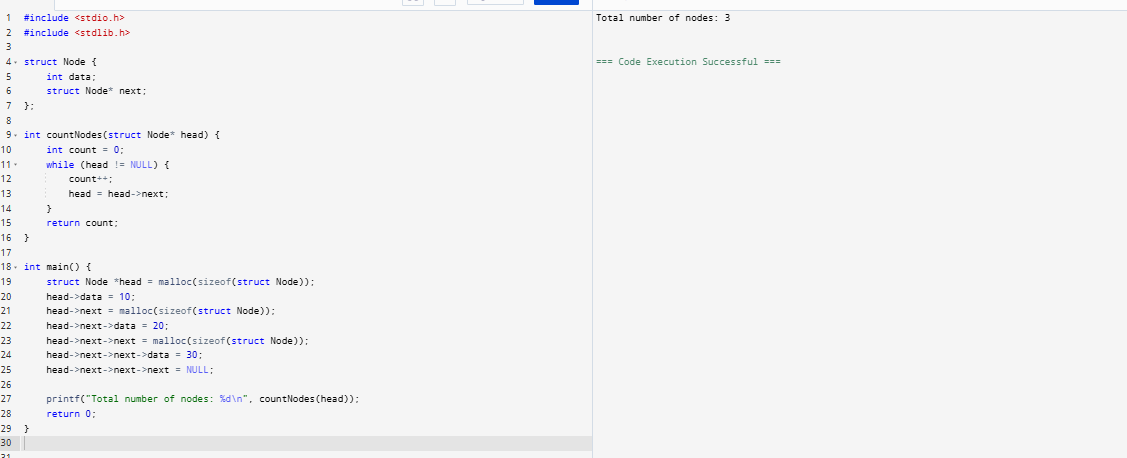
**Result :**

The program was successfully executed and the array was sorted using the Quick Sort algorithm.

25. Print Number of Nodes in a Linked List

**Aim:**

To count and display the number of nodes in a singly linked list.



**Result :**

The program was successfully executed and the total number of nodes in the linked list was counted and displayed.

**26. Given a 2D Matrix – Print Largest Element**

**Aim:**

To find and print the largest element in a given 2D matrix.



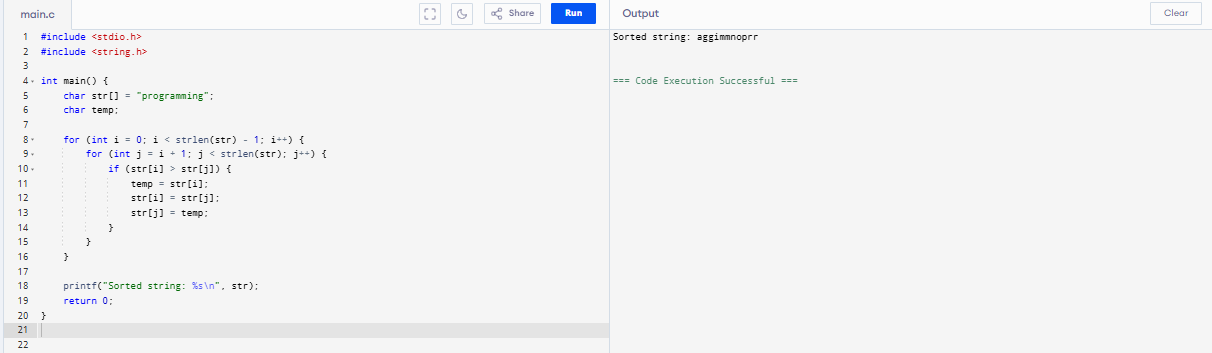
**Result:**

The program was successfully executed and the largest element in the given 2D matrix was found and displayed.

**27. Given a String – Sort in Alphabetical Order**

**Aim:**

To sort the characters of a string in alphabetical order.



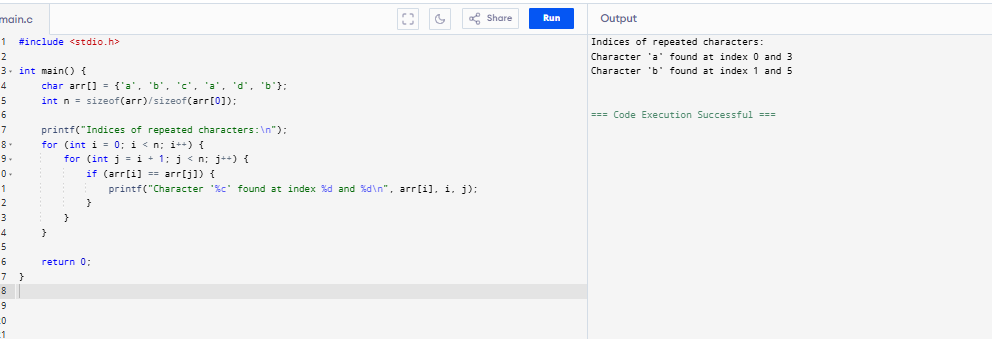
**Result:**

The program was successfully executed and the characters of the string were sorted alphabetically.

**28. Print Index of Repeated Characters in an Array**

**Aim:**

To identify and print the indices of repeated characters in an array.



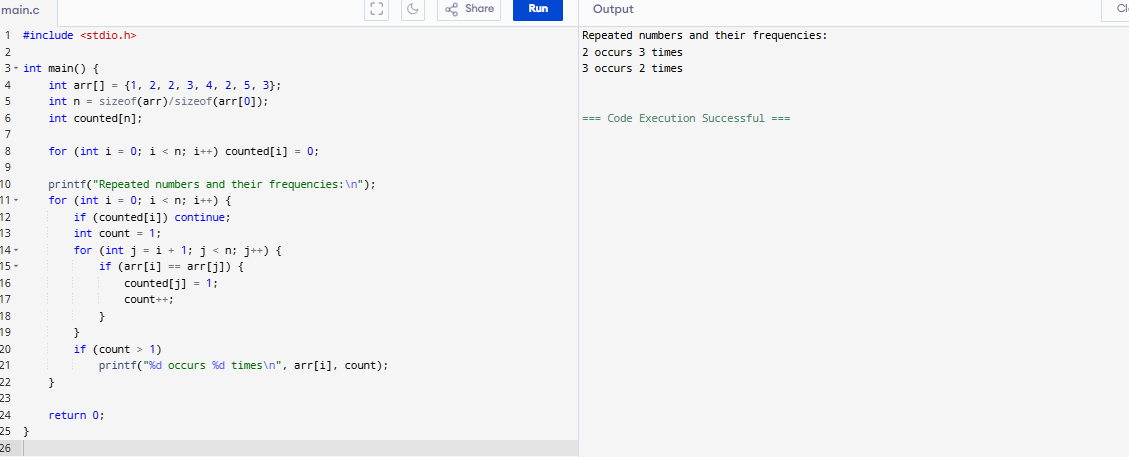
**Result :**

The program was successfully executed and the indices of the repeated characters in the array were identified and displayed.

**29. Print Frequently Repeated Numbers Count from an Array**

**Aim:**

To count and display the frequency of repeated numbers in an array.



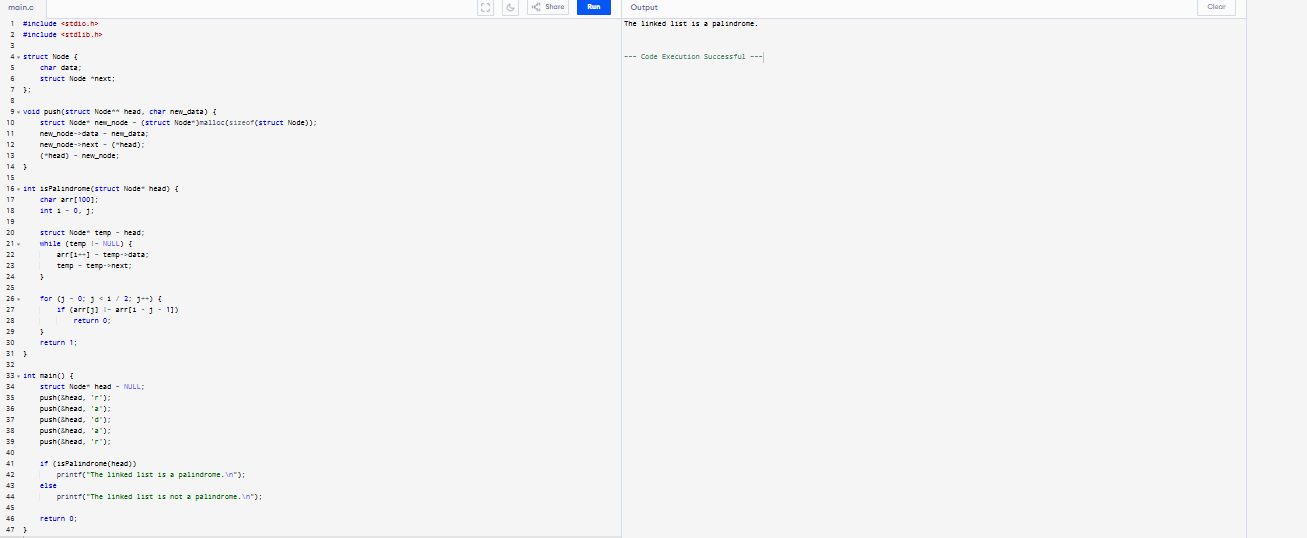
**Result in Words:**

The program was successfully executed and the frequency count of repeated numbers in the array was displayed.

**30. Palindrome Using Singly Linked List (SLL)**

**Aim:**

To check whether a singly linked list represents a palindrome.



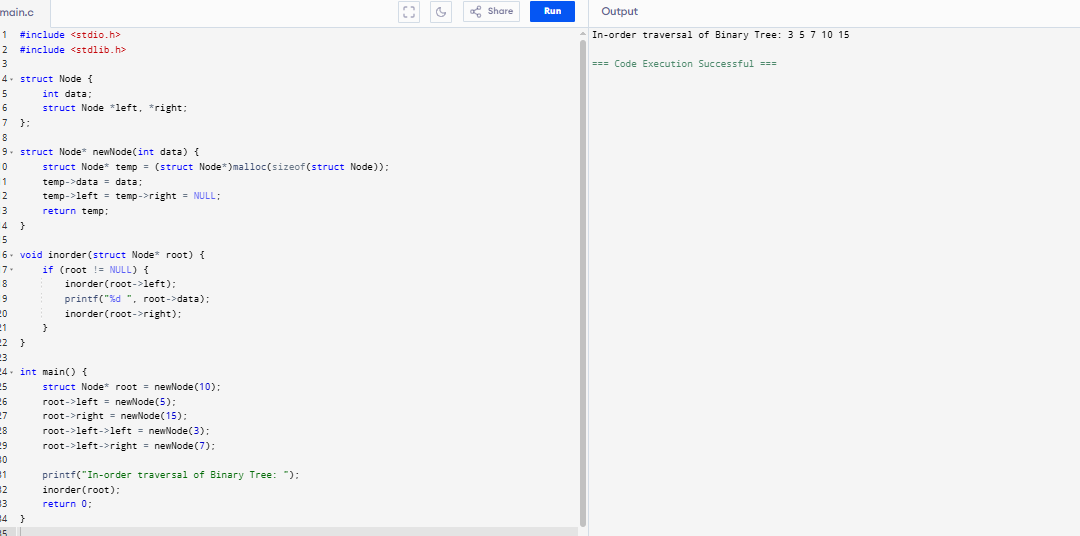
**Result :**

The program was successfully executed and it verified whether the linked list is a palindrome by checking the order of elements from both ends.

**31. Binary Tree**

**Aim:**

To implement a binary tree and perform in-order traversal.



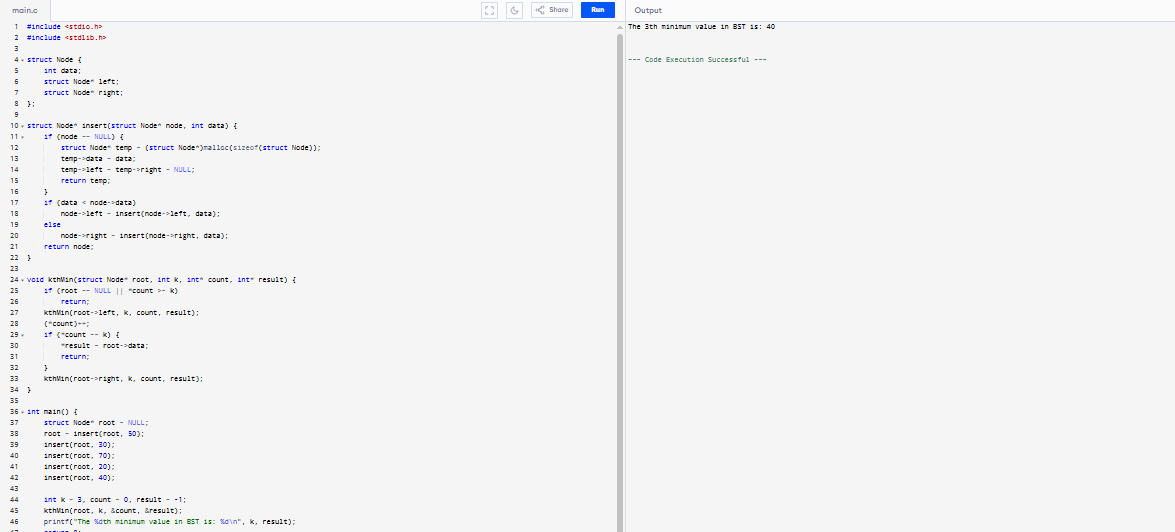
**Result :**

The program was successfully executed and a binary tree was created. Its in-order traversal was performed and displayed.

**32. BST – kth Minimum Value**

**Aim:**

To find the kth minimum value in a Binary Search Tree (BST).



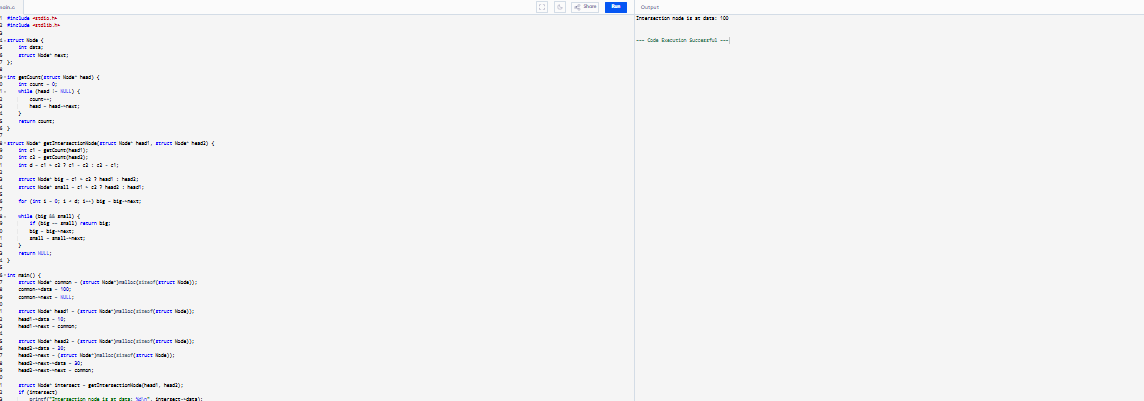
**Result :**

The program was successfully executed and the kth minimum value in the BST was identified and displayed.

**33. Intersecting Singly Linked Lists**

**Aim:**

To find the intersection node of two singly linked lists.



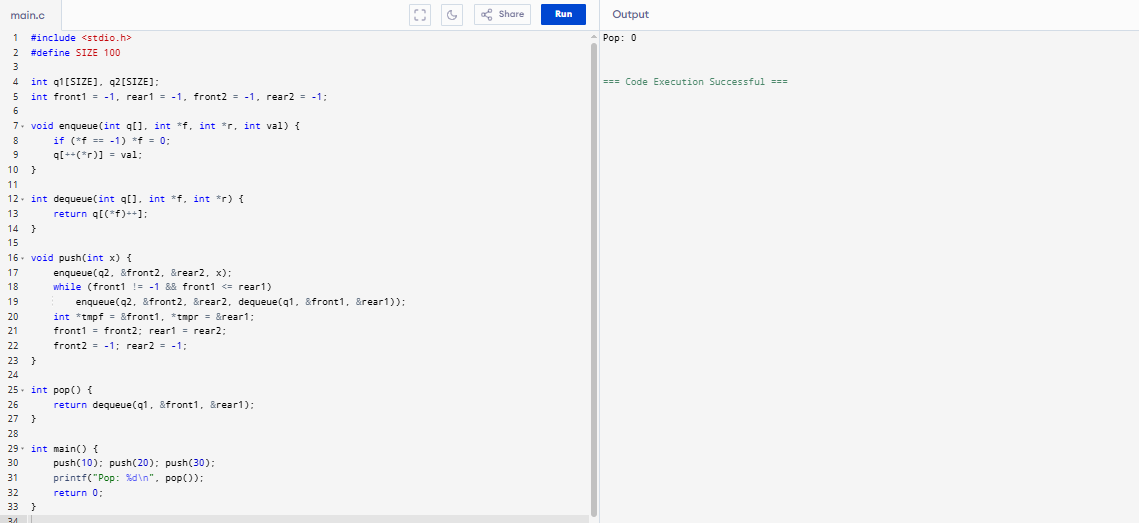
**Result :**

The program was successfully executed and the intersection point of two singly linked lists was found and displayed.

**34. Stack Using Two Queues**

**Aim:**

To implement a stack using two queues.



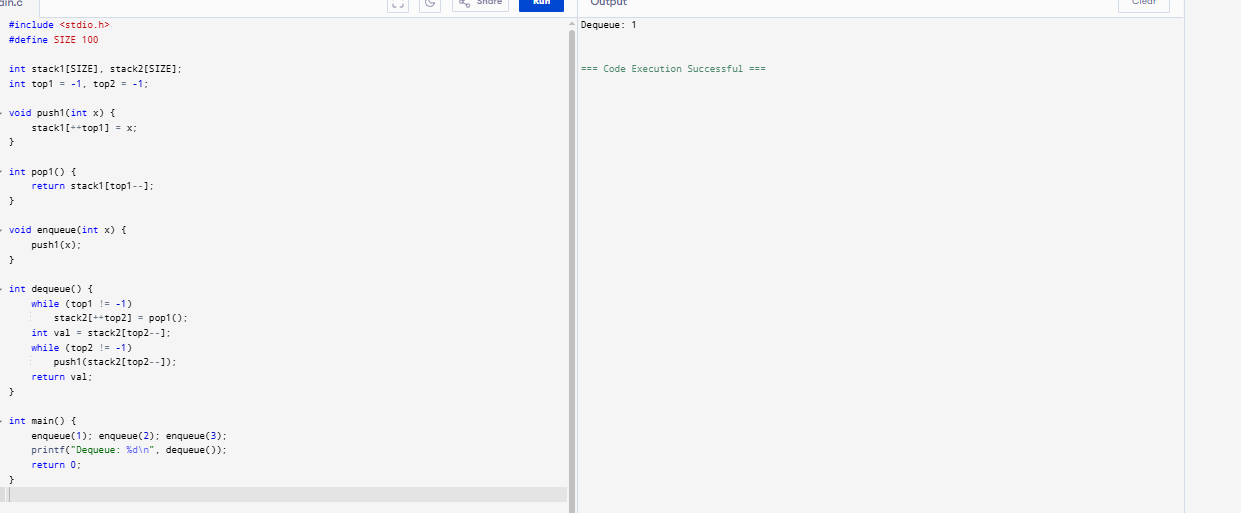
**Result :**

The program was successfully executed and a stack was implemented using two queues with correct LIFO behaviour.

**35. Queue Using Two Stacks**

**Aim:**

To implement a queue using two stacks.



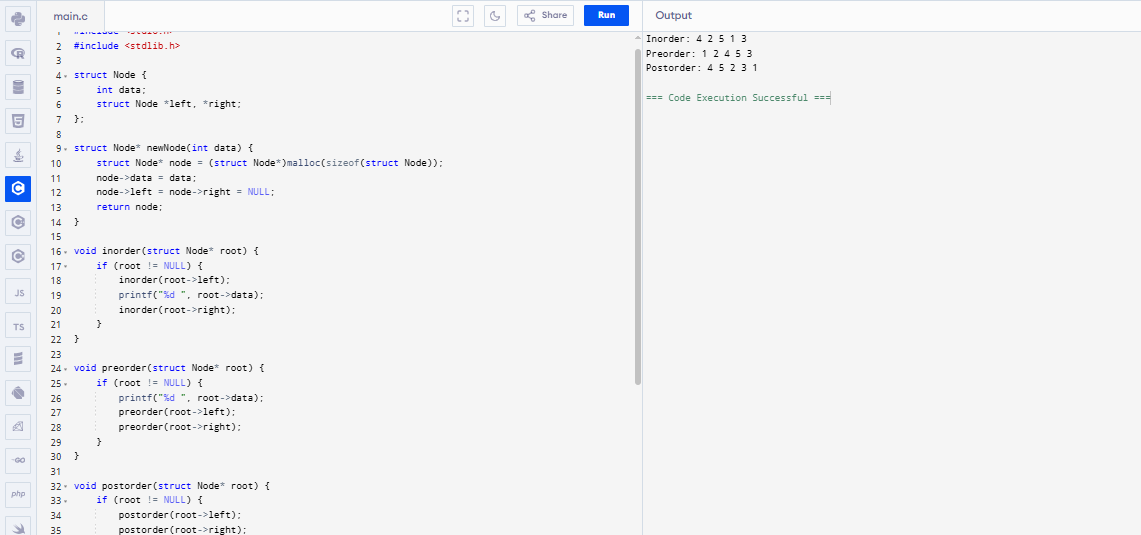
**Result**

The program was successfully executed and a queue was implemented using two stacks with correct FIFO behaviour.

**36. Tree traverse**

**Aim:**

To perform inorder, preorder, and postorder traversal on a binary tree.





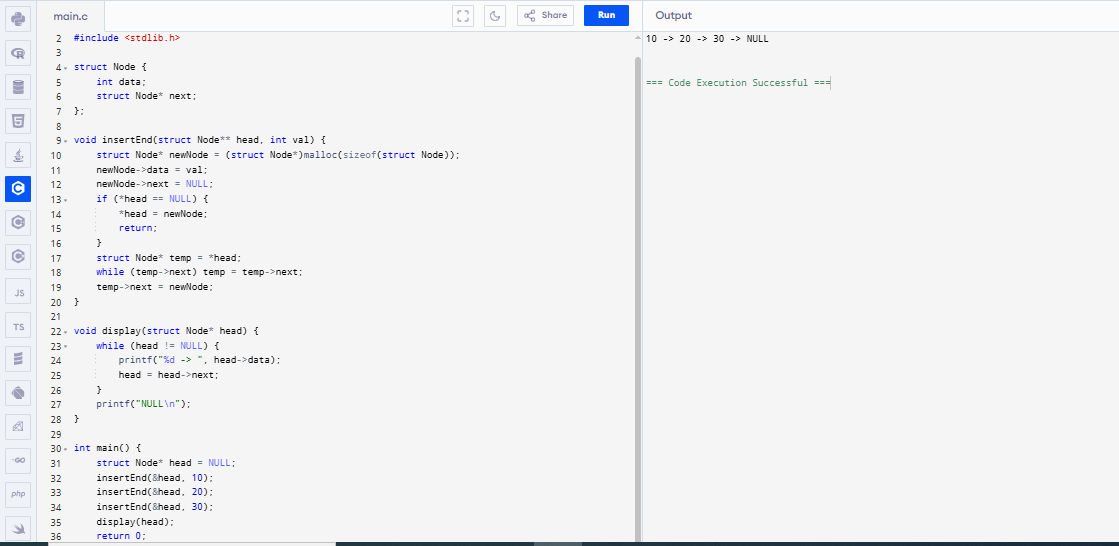
**Result :**

The program was successfully executed and the binary tree was traversed using inorder, preorder, and postorder traversal methods.

37 linked list – Insertion

**Aim:**

To insert nodes at the beginning, end, and a given position in a singly linked list.

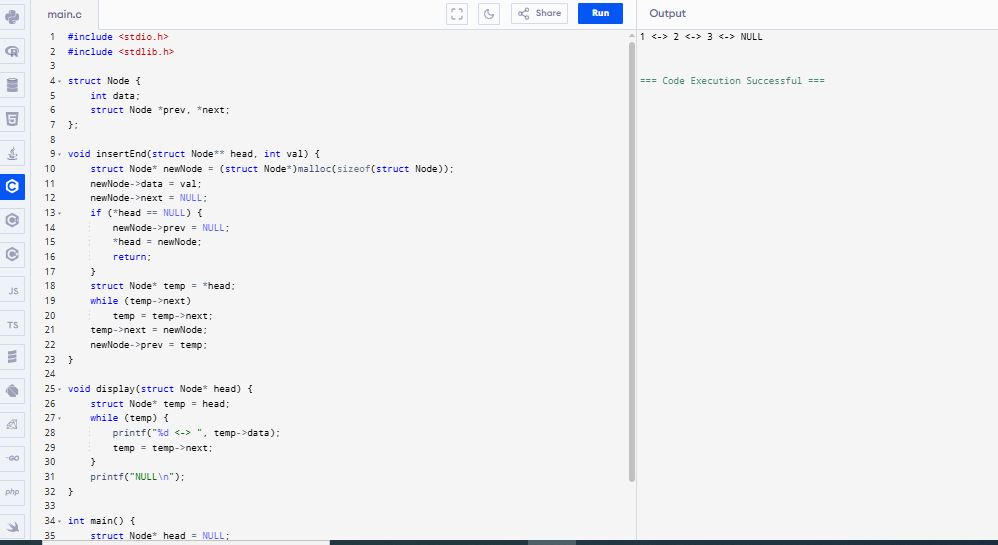


**Result** The program was successfully executed and nodes were inserted at the end of a singly linked list and displayed in correct order.

**38. Bidirectional (Doubly) Linked List**

**Aim:**

To implement a doubly linked list and perform insertion and display operations.



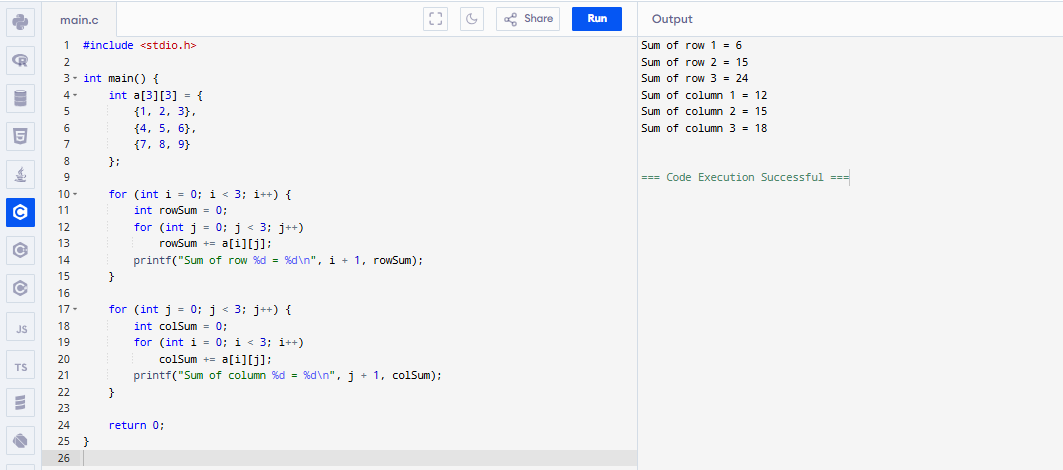
**Result:**

The program was successfully executed and a doubly linked list was created. Nodes were inserted and displayed in both directions.

**39. Sum of Row and Column – 2D Array**

**Aim:**

To compute and print the sum of each row and column in a 2D matrix.



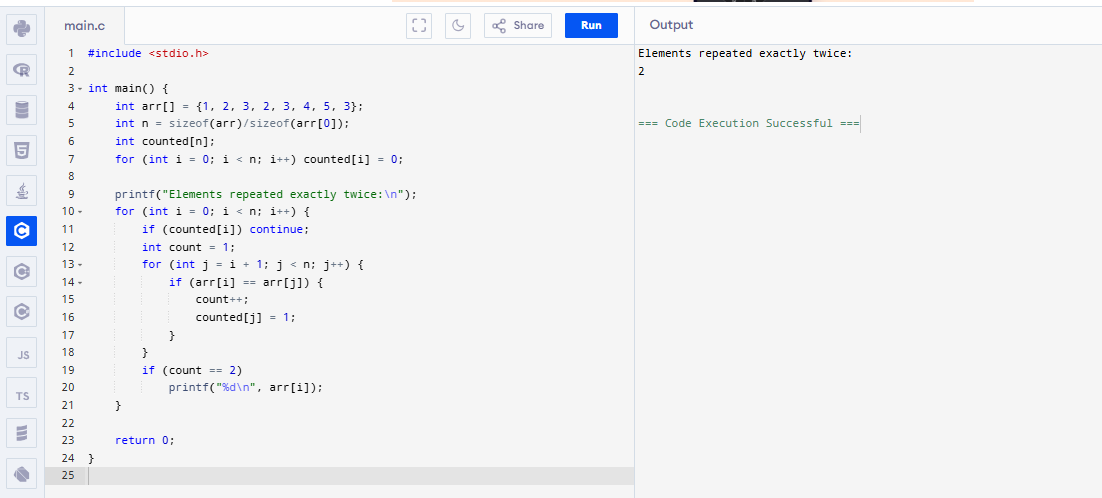
**Result :**

The program was successfully executed and the sum of each row and each column in the matrix was calculated and displayed.

**40. Elements Repeated Twice – Array**

**Aim:**

To find and display all elements that are repeated exactly twice in an array.



**Result :** The program was successfully executed and elements that occurred exactly twice in the array were identified and displayed**.**