**Data Structures and Algorithms: Programming Assignment**

**Course:** MBA. Tech CE (Div-A)– Semester III  
**Subject Code:** 702CO1C001  
**Instructor:** Dr. Nitin Choubey  
**Submission Deadline:** 01st November 2025  
**Mode of Submission:** Submit GitHub repo link or PDF with screenshots and code as submission for this.

**Objective**

To strengthen your understanding of core data structures and algorithms by solving real-world problems on competitive programming platforms. This assignment will help you develop problem-solving skills, optimize code, and prepare for technical interviews.

**Platforms**

You may use any of the following platforms:

1. LeetCode
2. CodeChef
3. HackerRank
4. GeeksforGeeks Practice

**Assignment Task: Data Structures Practice**

Solve any **two problems** from the mentioned categories,

1. Array
2. Linked List
3. Stack
4. Queue

**(Note: if you are chosen first problem from the Array, then the second problem should be from the other three categories)**

**Submission Guidelines**

* Create a GitHub repository named DSA\_Assignment\_YourName
* For each problem, include:
  + Problem link
  + Your code (with comments)
  + Screenshot of successful submission
  + Brief explanation of your approach (2–3 lines)
* Alternatively, compile all of the above into a single PDF and upload it to MS Teams.

**Evaluation Criteria**

* Completion of required problems
* Code correctness and efficiency
* Clarity of explanations
* Proper submission format

Name: Tilak Chauhan

Roll No: N240

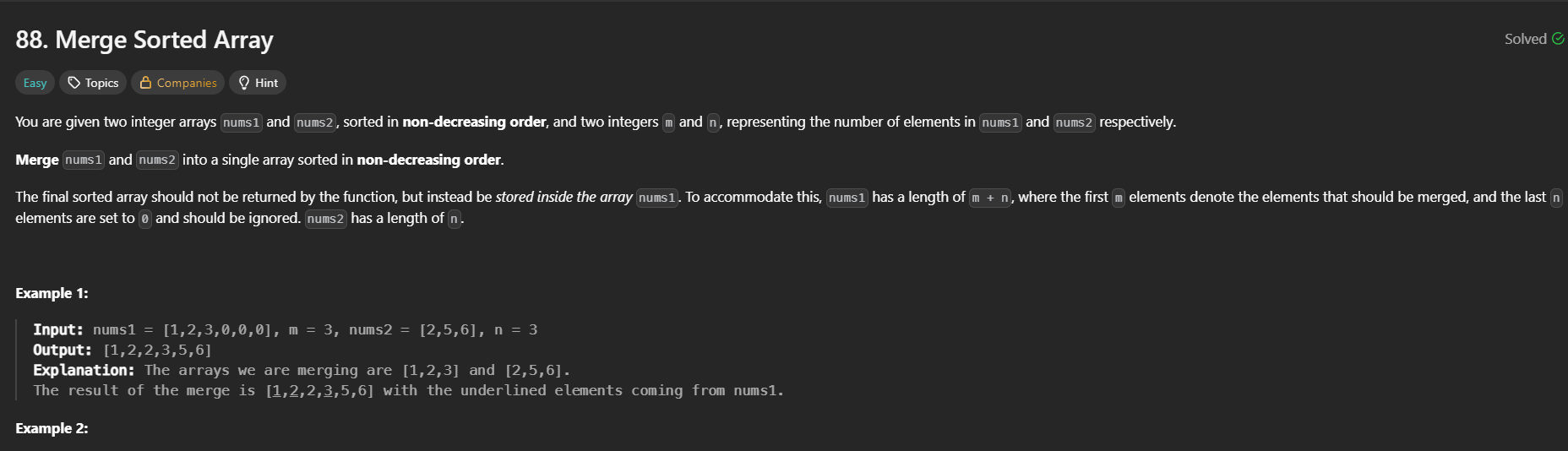
Batch: A1

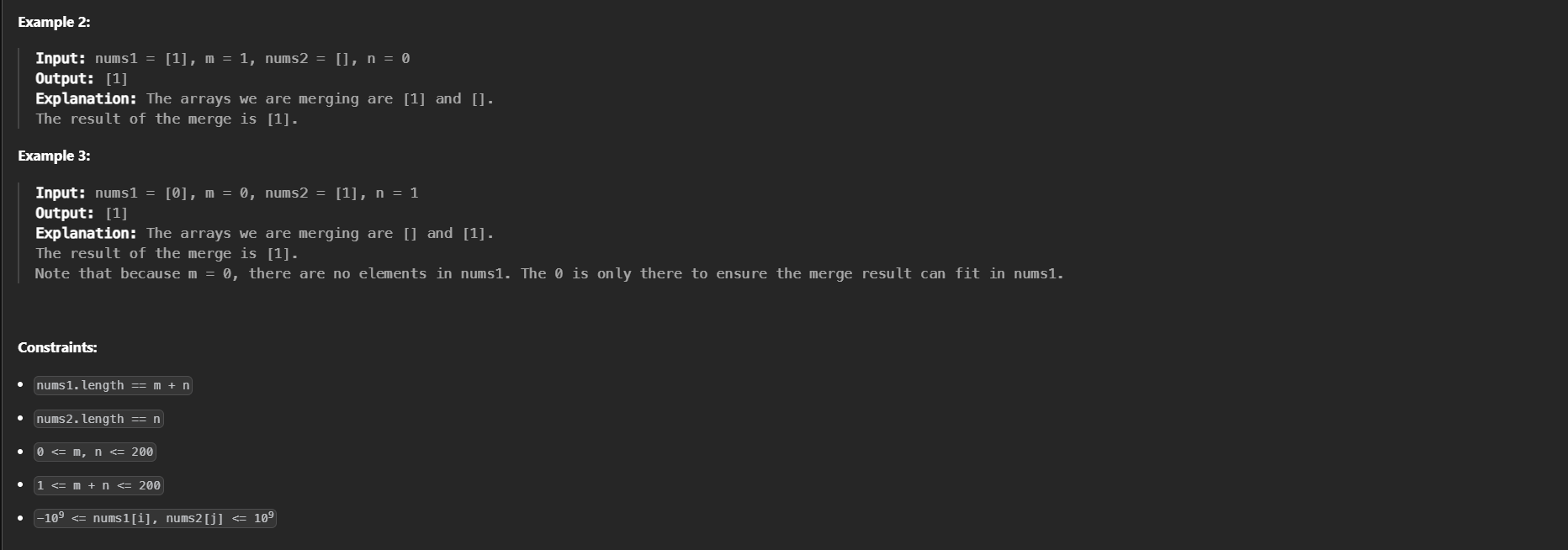
Class: MBA-TECH(CE)

Github link: <https://github.com/chadda26/DSA_Assignment_Tilak>

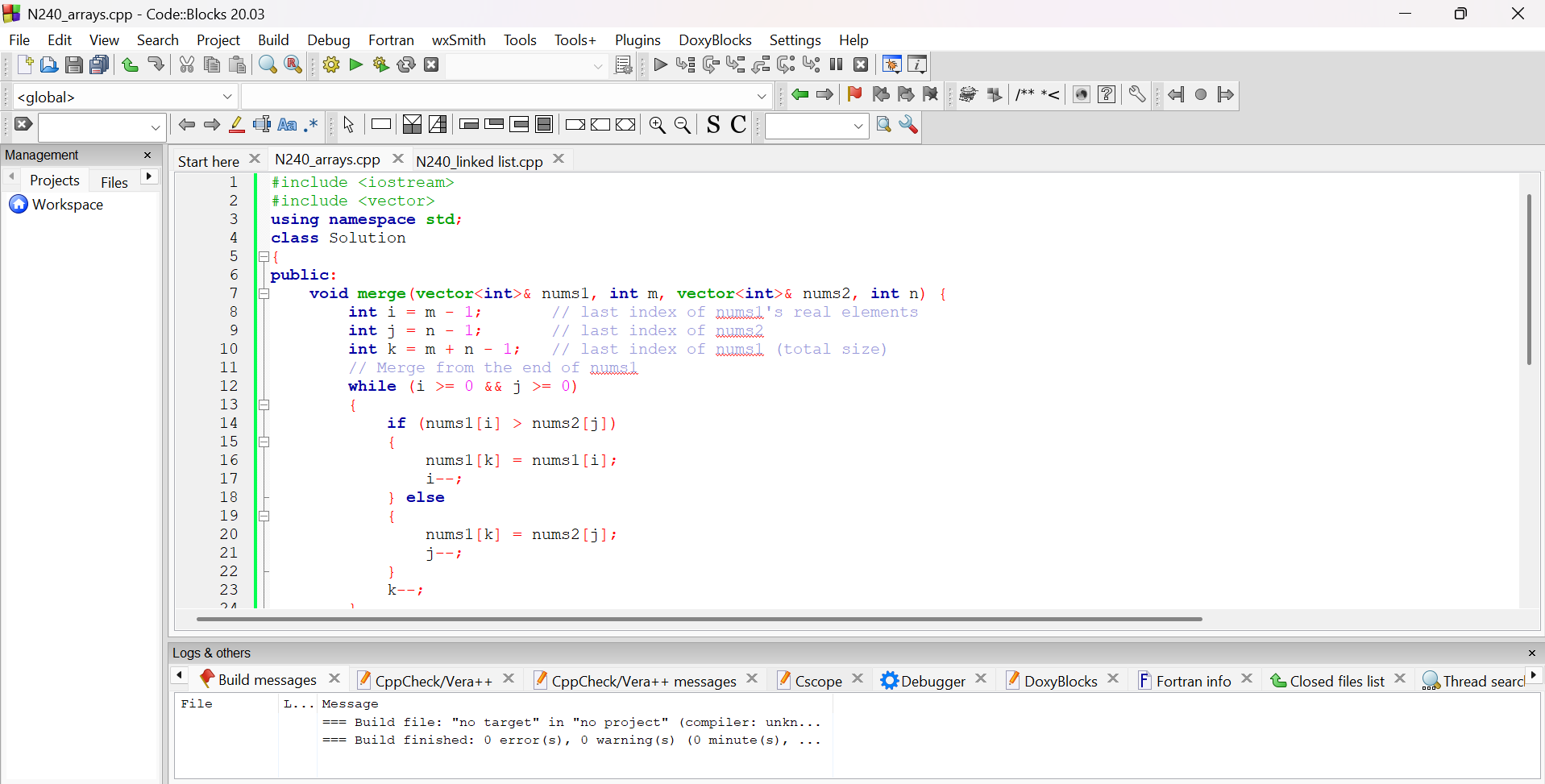
1. Arrays: <https://leetcode.com/problems/merge-sorted-array/>

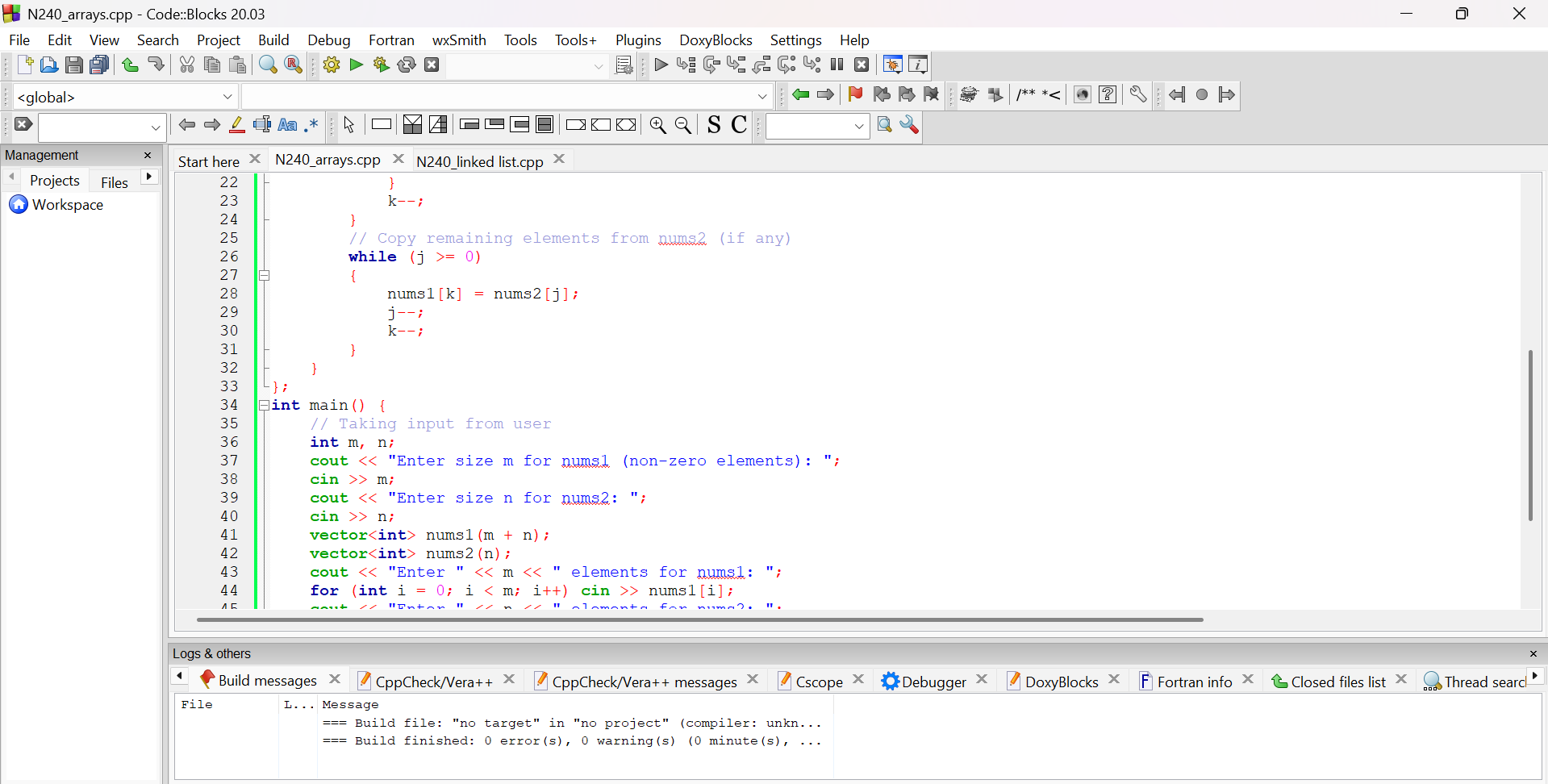
**PROBLEM STATEMENT**

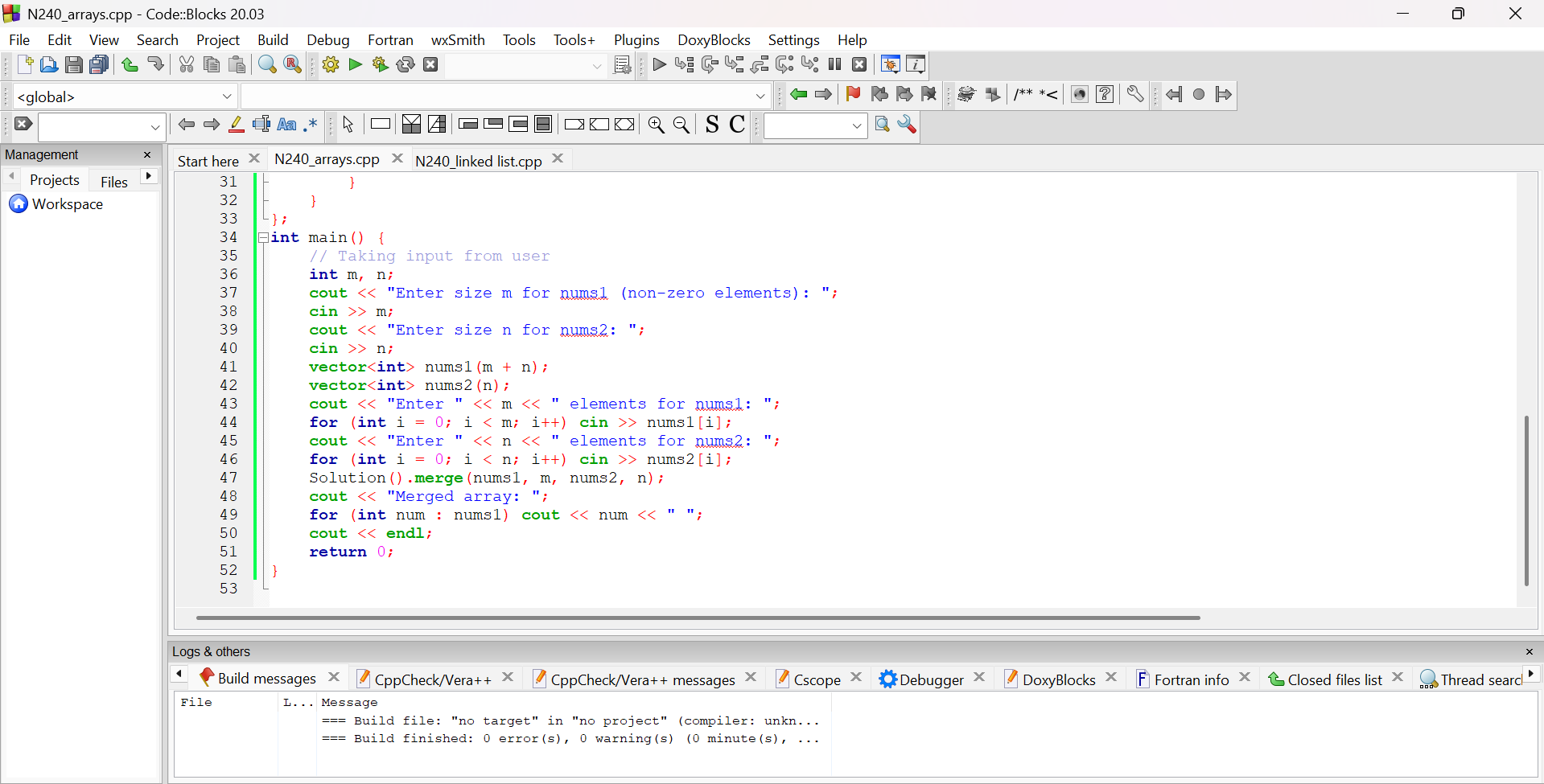




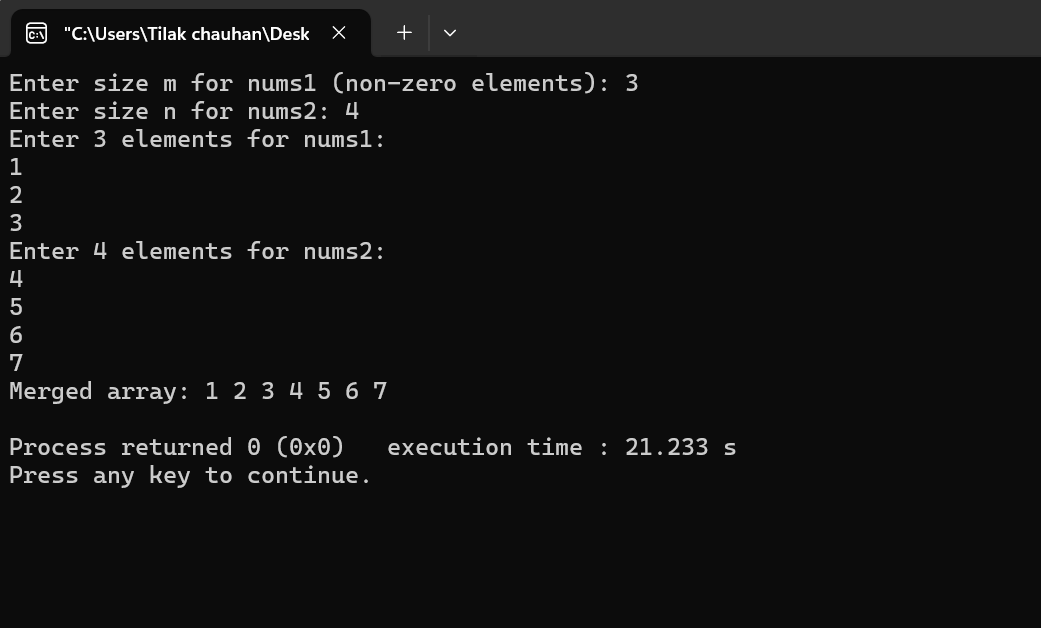
**CODE**







**OUTPUT**

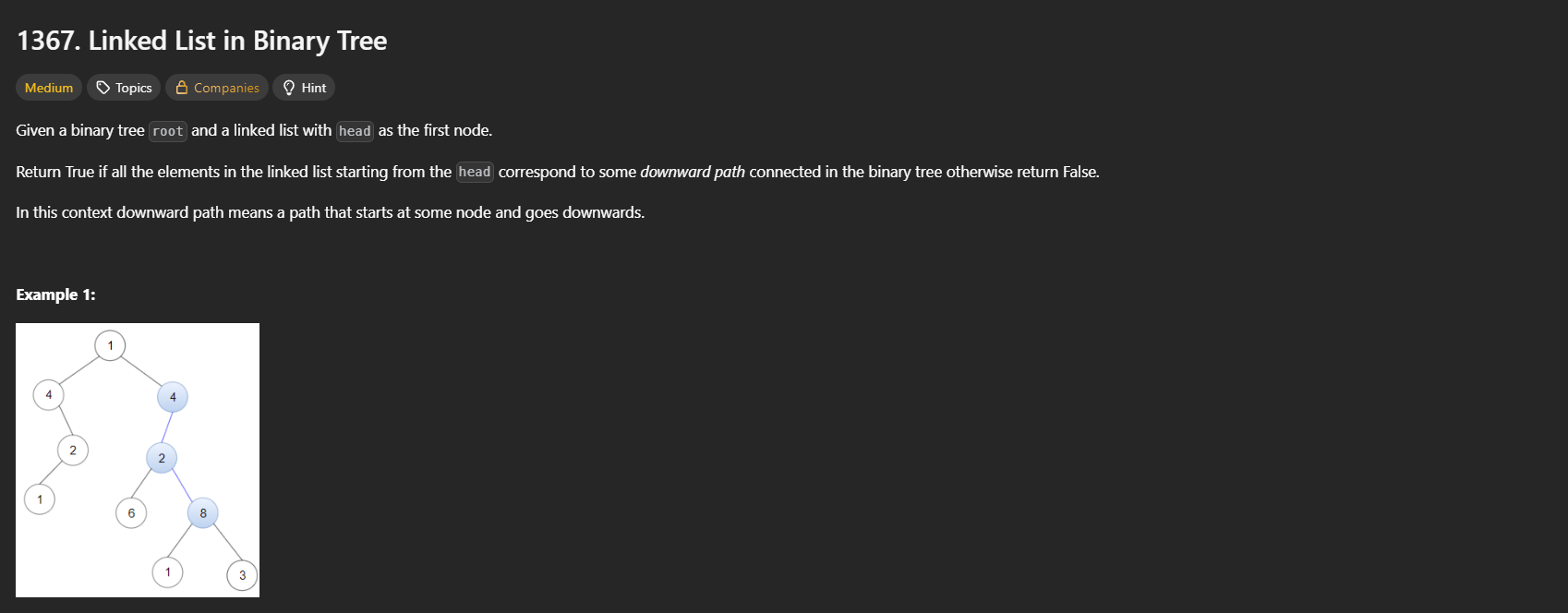


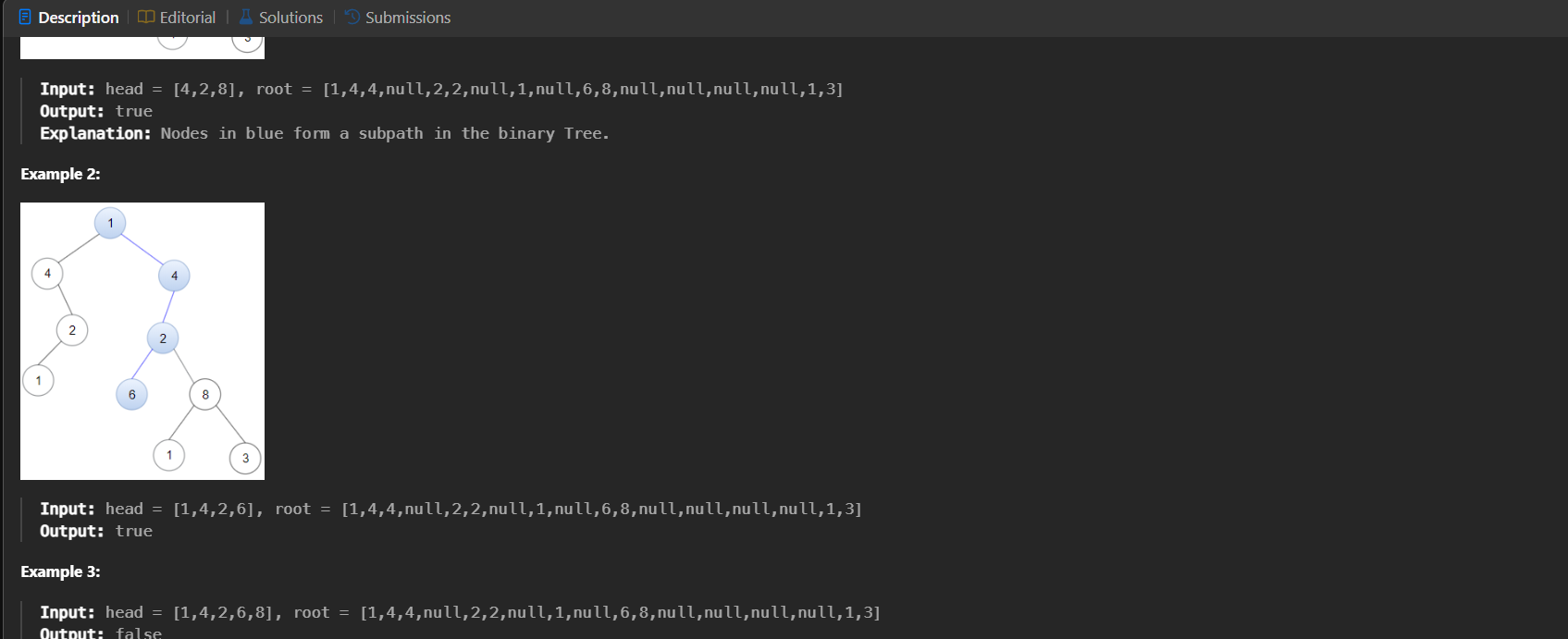
**BRIEF DESCRIPTION ABOUT THE APPROACH**

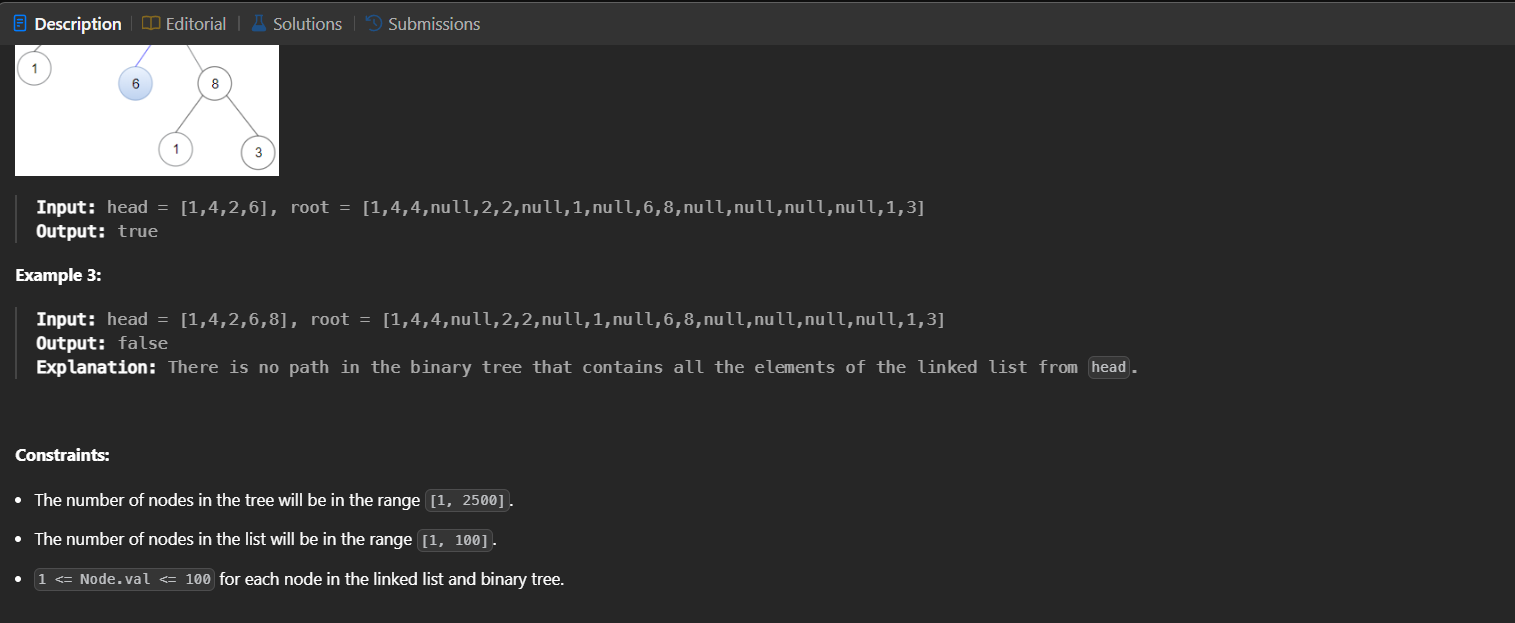
The approach for merging two sorted arrays is to start from the **end** of both arrays to avoid overwriting elements in nums1. Three pointers are used — one for the end of nums1’s valid part, one for nums2, and one for the final merged position. The larger element between the two arrays is placed at the current end position, and pointers move backward. This continues until all elements are merged, achieving an **O(m + n)** time complexity and **O(1)** space complexity.

1. Linked List : <https://leetcode.com/problems/linked-list-in-binary-tree/description/>

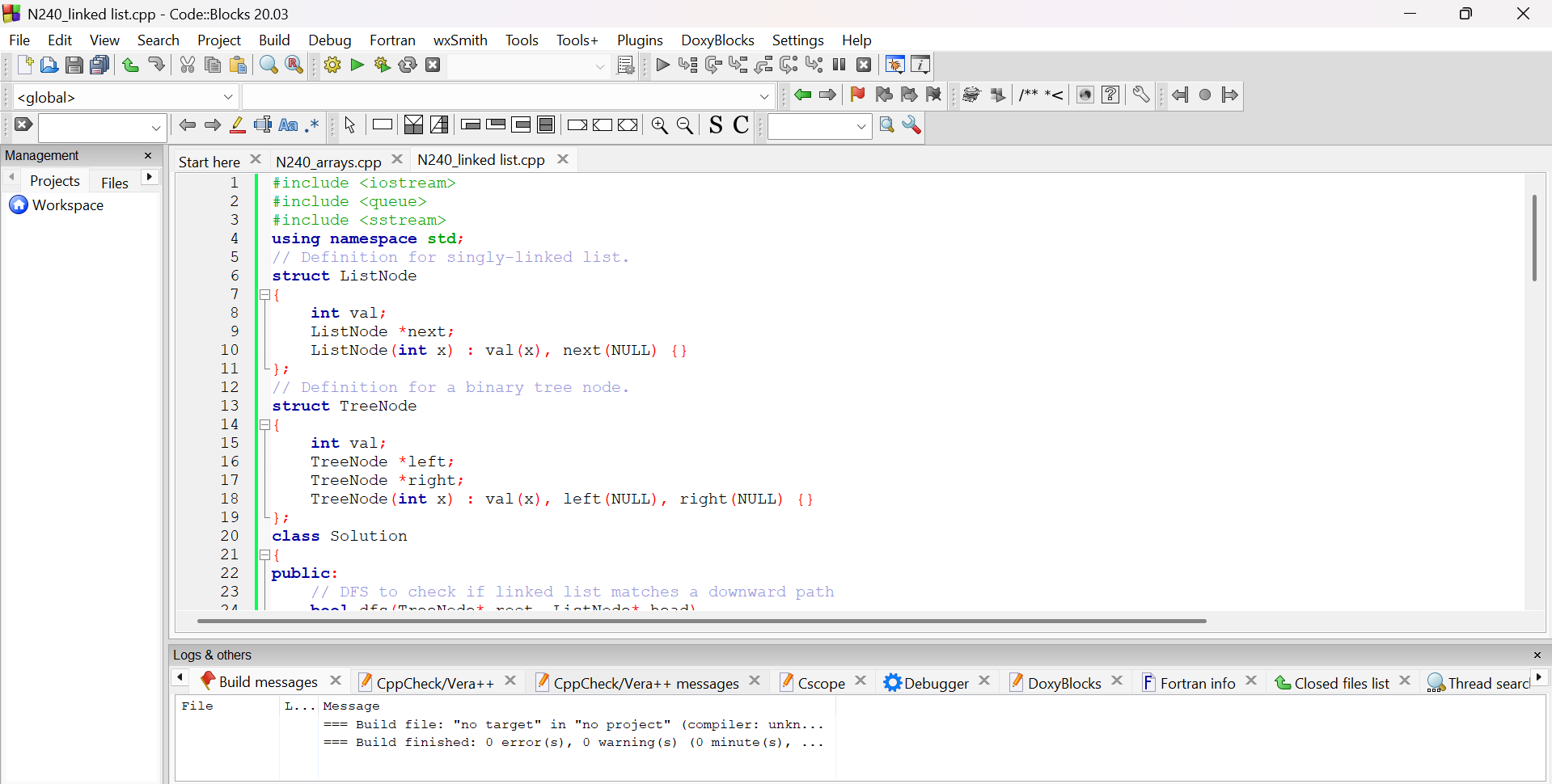
**PROBLEM STATEMENT**

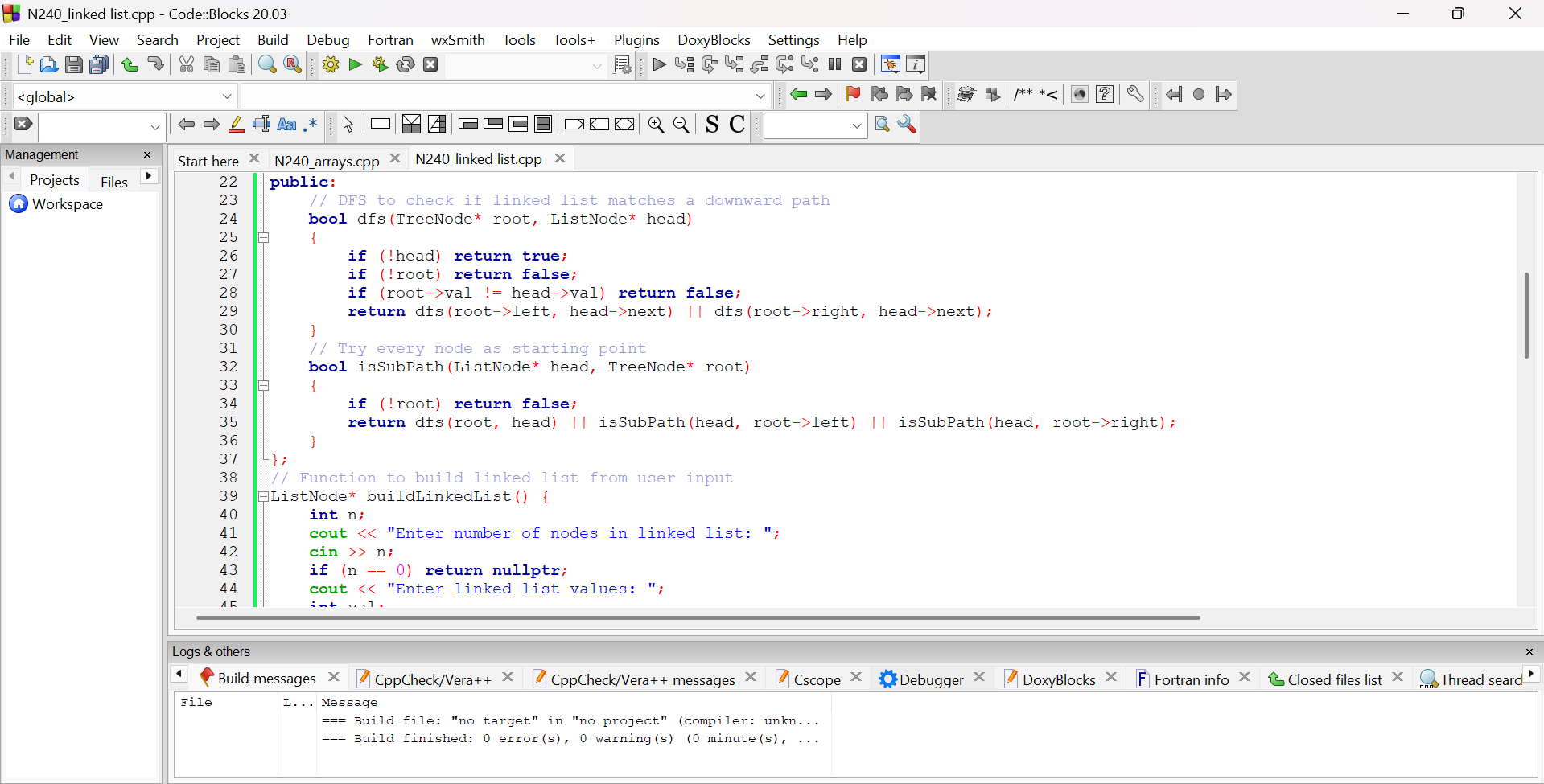


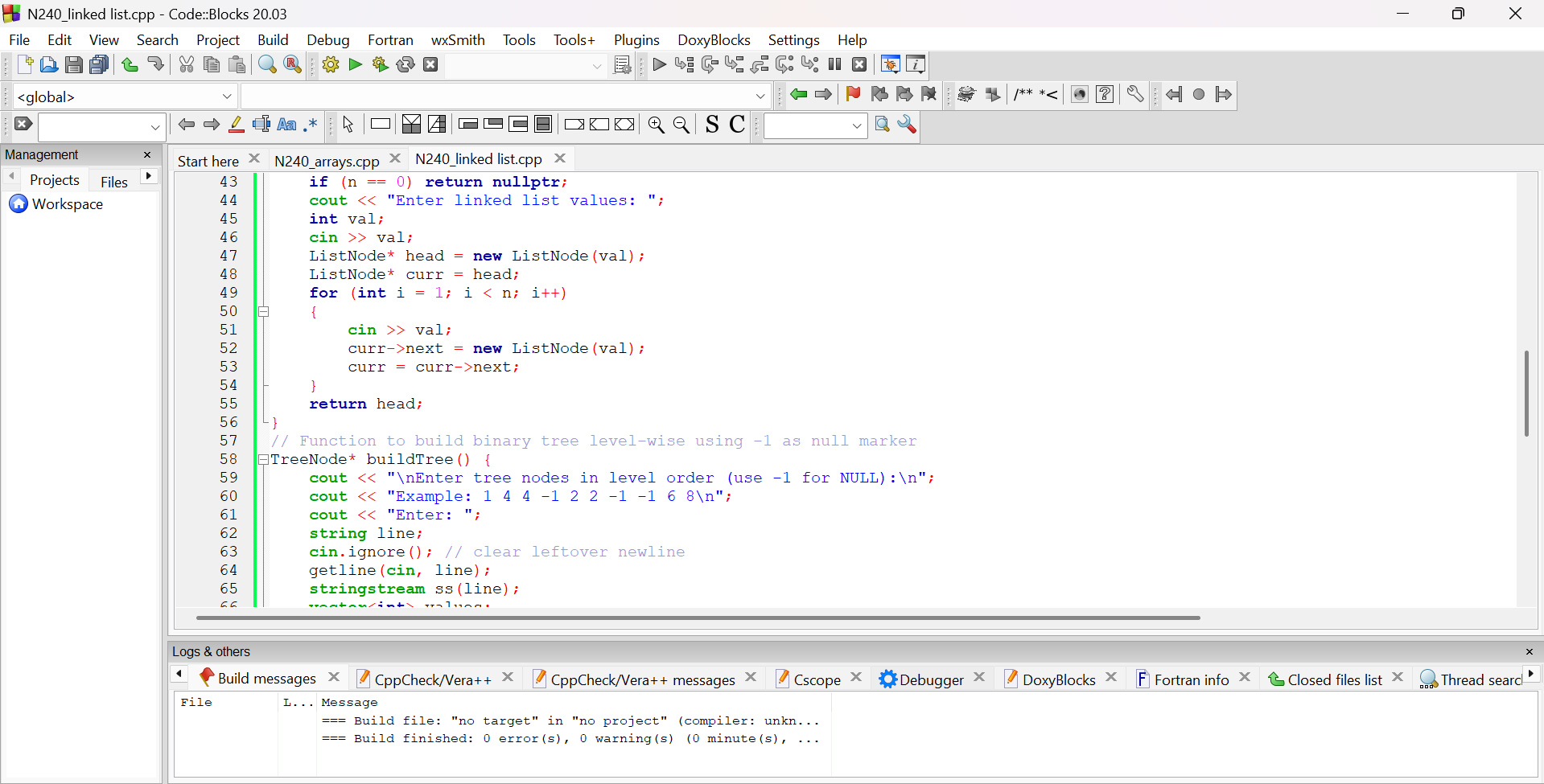


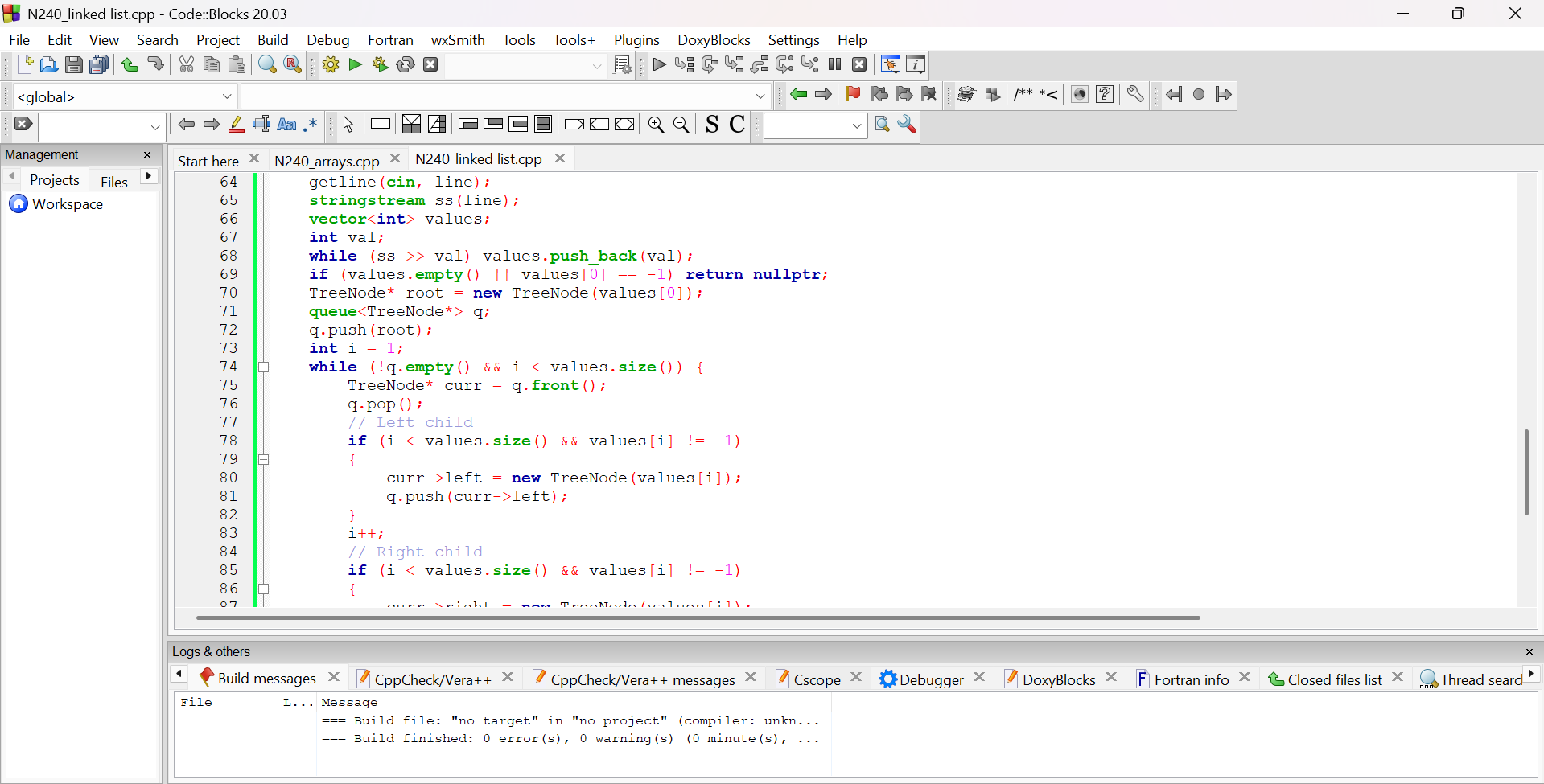


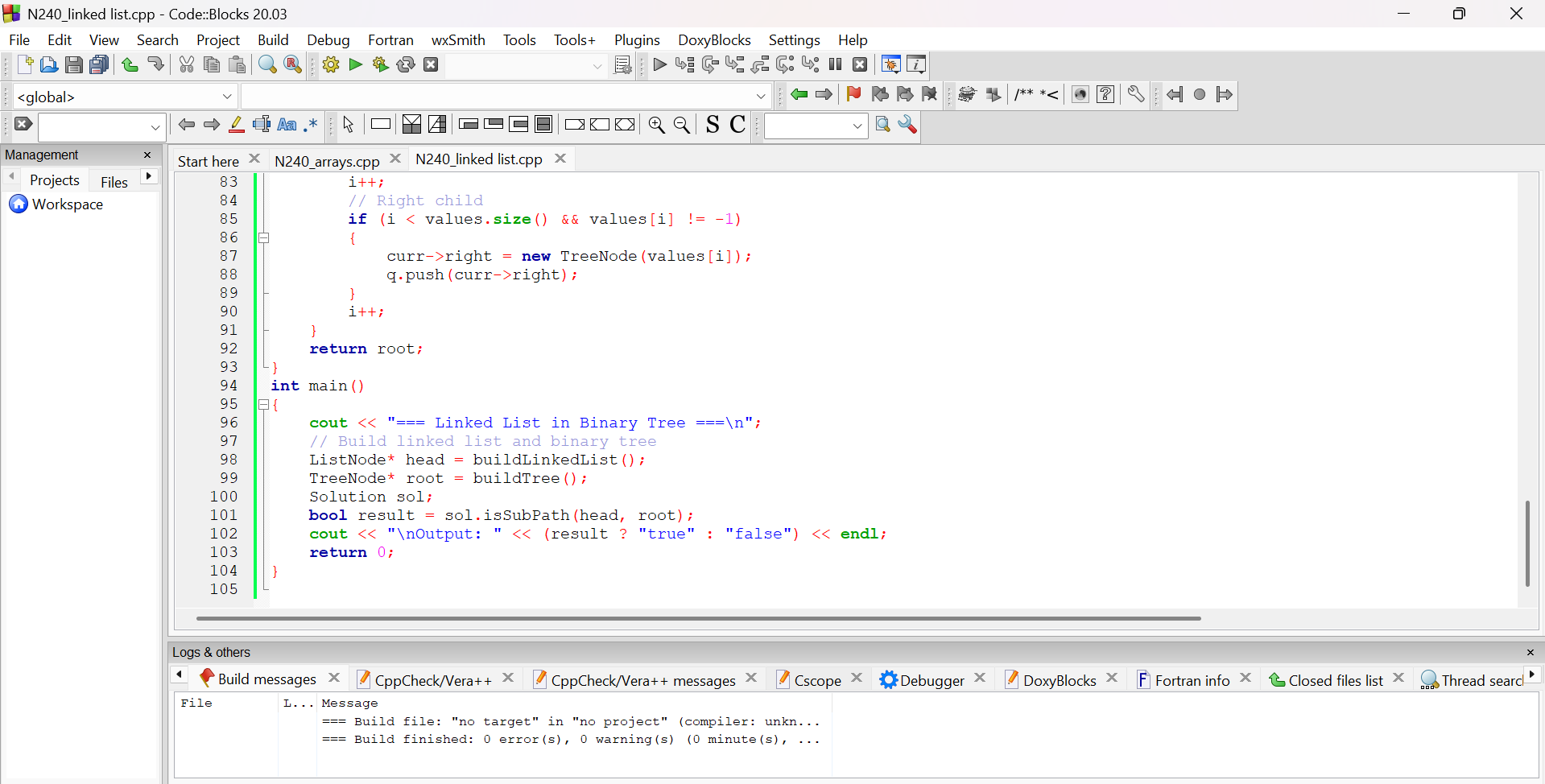
**CODE**



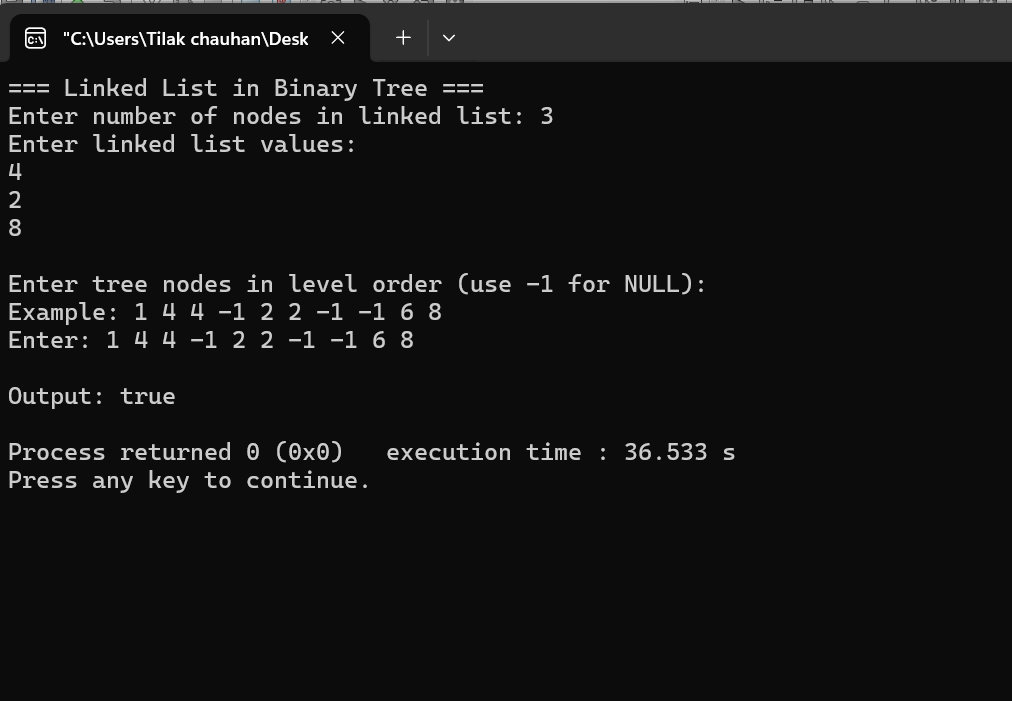








**OUTPUT**

****

**BRIEF DESCRIPTION ABOUT THE APPROACH**

The approach for the **“Linked List in Binary Tree”** problem relies on using **Depth-First Search (DFS)** to determine if the sequence of values in the linked list matches any downward path in the binary tree. Starting from each node in the tree, we check whether its value matches the head of the linked list. If it does, we recursively continue the comparison along both the left and right child nodes for the next linked list elements. If the entire linked list is successfully matched along a path, we return true. Otherwise, the algorithm continues checking from other tree nodes. This exhaustive search ensures all possible paths are considered, resulting in a time complexity of **O(N × M)**, where N is the number of tree nodes and M is the number of linked list nodes.