**Name**- Manasvi Mhadev Patil

**Class** - SY IT-C

**Roll No.** – 03

**PRN** – 12211105

**Subject** - Advanced Data Structures

**Aim** - Implement the program in C or C++: Create BT/BST and insert element of user choice, Recursive – Inorder, Preorder, Postorder Tree Traversal

**Theory**:

**About Trees**:

Trees are hierarchical data structures that represent a non-linear and non-primitive form of organization. They are composed of interconnected nodes, where each node maintains references to other nodes. The nodes are divided into three sections, with the middle portion holding the data, while the left and right sections store the addresses of subsequent or child nodes. In cases where no children exist, a null value is stored. The highest node in the tree is referred to as the root.

Some applications of trees:

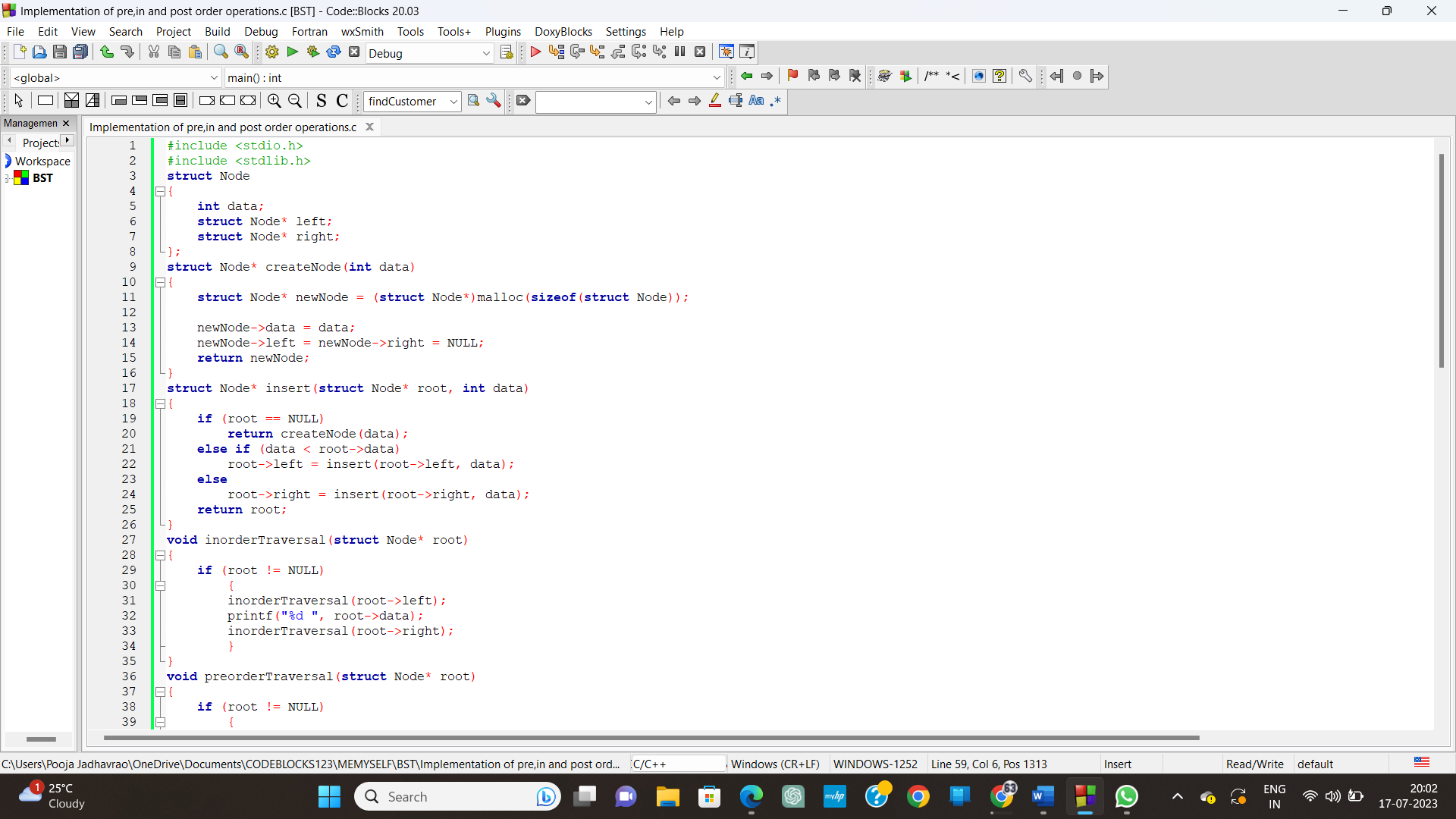
1. File Systems
2. Organization Structures
3. Compiler Design (Syntax trees)
4. Network Routing Algorithms
5. Decision Trees (Machine Learning)

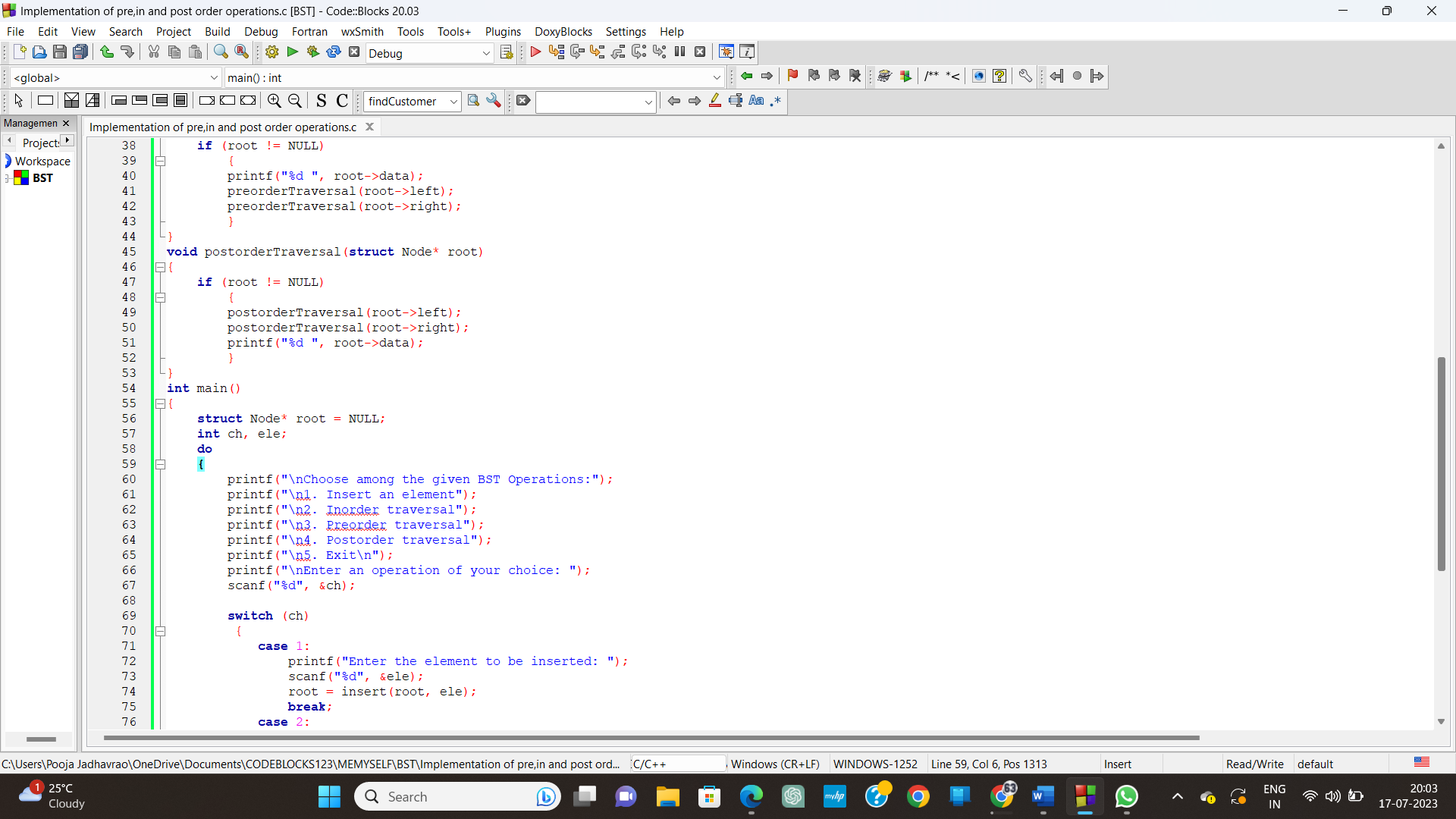
Various Operations performed on trees are:

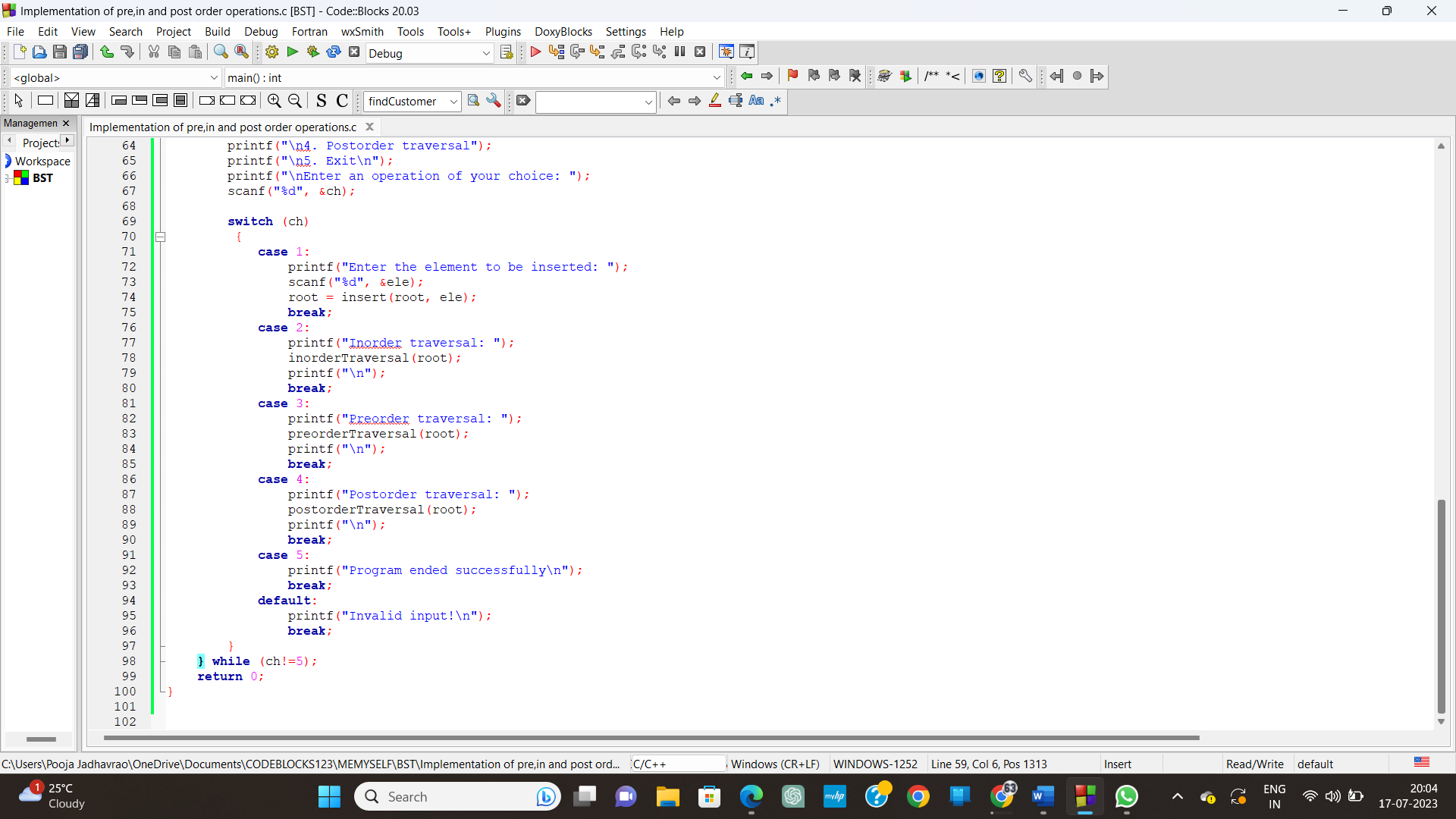
1. **Insertion of elements**:Insertion of new data in trees involves adding a new node with a specific value to the existing tree structure. Starting from the root node, the new node's value is compared with the current node's value. If the value is less, the insertion process continues in the left subtree, if it is greater, it continues in the right subtree. This comparison and traversal process continues until an appropriate position is found where the new node can be inserted, ensuring that the tree maintains its hierarchical ordering
2. **Inorder Traversal**:Starting from the leftmost node, we traverse its left subtree, then visit the node itself, and traverse its right subtree. In other words, for every node, we first visit its left child, then the node itself, and finally its right child. In terms of tree traversal, the inorder traversal visits the nodes in ascending order when the tree represents a sorted set of values. It is commonly used to retrieve elements from a binary search tree in sorted order.
3. **Preorder Traversal**:In this traversal, we first visit the node itself, traverse its left subtree, and traverse its right subtree. In other words, for every node, we first visit the node itself, then its left child, and finally its right child. Preorder traversal is often used to create a copy of the tree or to perform certain operations on the nodes before their children. It can be helpful for constructing an expression tree or for evaluating prefix expressions.
4. **Postorder Traversal**:

It involves traversing the left subtree first, then the right subtree, and finally visiting the node itself. In other words, for every node, we first visit its left child, then its right child, and finally the node itself. Postorder traversal is often used when we need to perform operations on the child nodes before processing the parent node. It can be helpful for tasks like deleting a tree, as we typically free the memory allocated for the child nodes before deleting the parent node. Additionally, postorder traversal is useful for certain applications like expression tree evaluation, where the order of operations requires processing the operands before the operators.

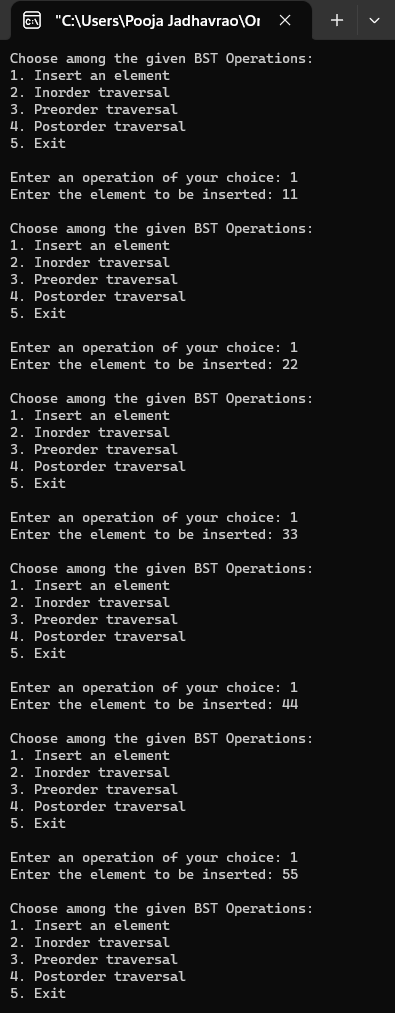
**Code**:

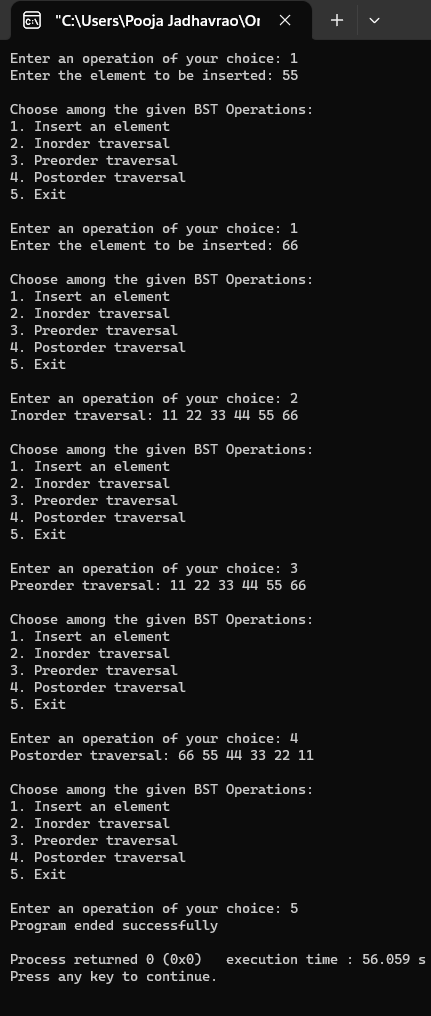






**Output**:





**Conclusion**:

The above menu-driven program demonstrates the implementation of a binary search tree in C. It allows users to interactively insert elements of their choice into the tree and perform various tree traversal operations such as inorder, preorder, and postorder traversals. By using a recursive approach, the program effectively navigates through the tree structure and prints the nodes in the desired order. Understanding and implementing tree operations like insertion and traversals are fundamental concepts in data structures and algorithms, with applications ranging from file systems to network routing and machine learning.