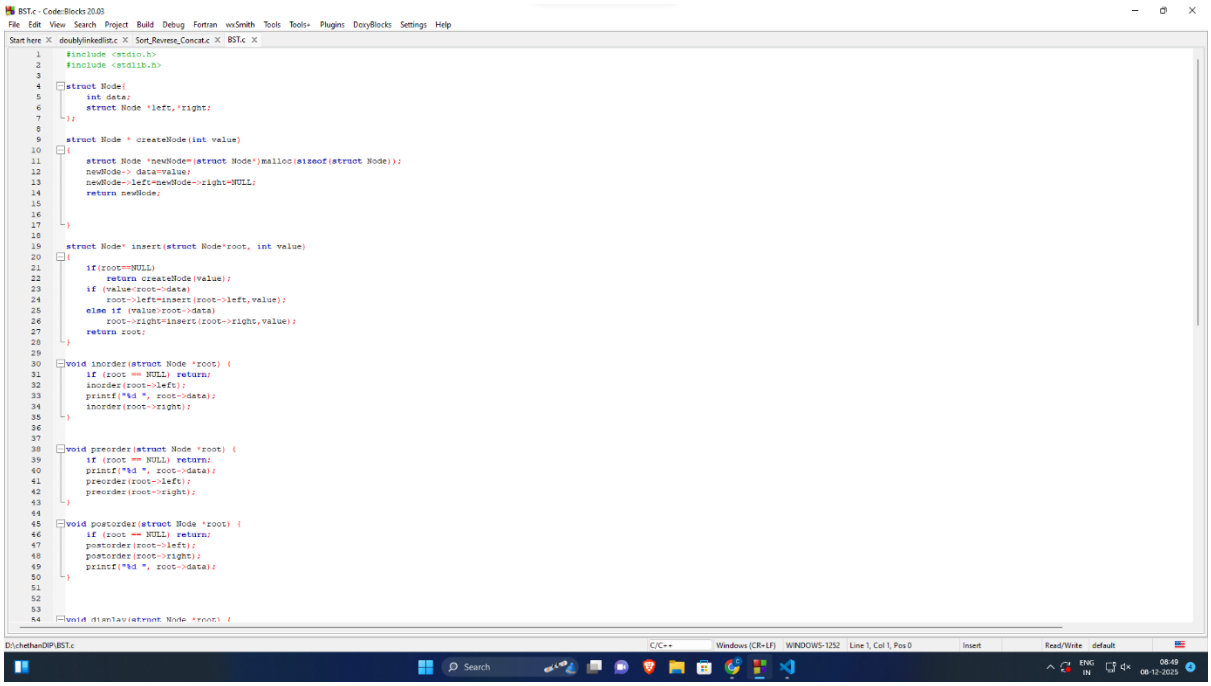
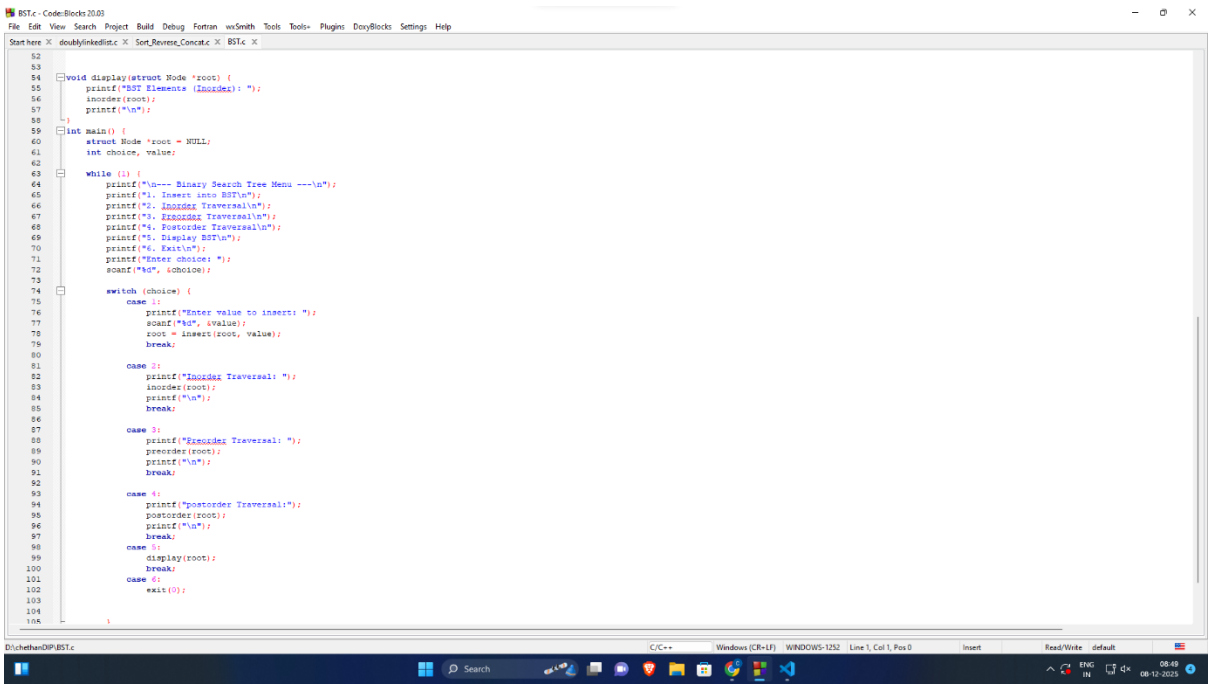
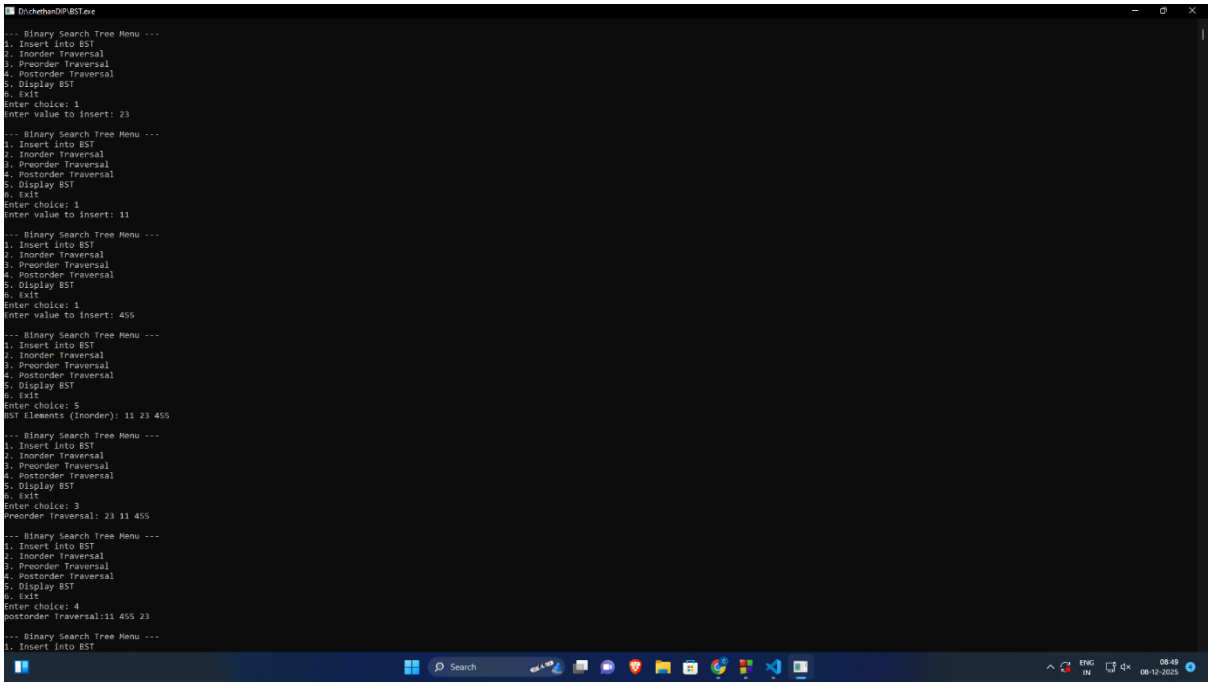


```c
#include <stdio.h>
#include <stdlib.h>

struct Node{
    int data;
    struct Node *left,*right;
};

struct Node * createNode(int value)
{
    struct Node *newNode=(struct Node*)malloc(sizeof(struct Node));
    newNode-> data=value;
    newNode->left=newNode->right=NULL;
    return newNode;


}

struct Node* insert(struct Node*root, int value)
{
    if(root==NULL)
        return createNode(value);
    if (value<root->data)
        root->left=insert(root->left,value);
    else if (value>root->data)
        root->right=insert(root->right,value);
    return root;
}

void inorder(struct Node *root) {
    if (root == NULL) return;
    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}


void preorder(struct Node *root) {
    if (root == NULL) return;
    printf("%d ", root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder(struct Node *root) {
    if (root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}


void display(struct Node *root) {
```



```c

void display(struct Node *root) {
    printf("BST Elements (Inorder): ");
    inorder(root);
    printf("\n");
}
int main() {
    struct Node *root = NULL;
    int choice, value;

    while (1) {
        printf("\n--- Binary Search Tree Menu ---\n");
        printf("1. Insert into BST\n");
        printf("2. Inorder Traversal\n");
        printf("3. Preorder Traversal\n");
        printf("4. Postorder Traversal\n");
        printf("5. Display BST\n");
        printf("6. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                root = insert(root, value);
                break;

            case 2:
                printf("Inorder Traversal: ");
                inorder(root);
                printf("\n");
                break;

            case 3:
                printf("Preorder Traversal: ");
                preorder(root);
                printf("\n");
                break;

            case 4:
                printf("postorder Traversal:");
                postorder(root);
                printf("\n");
                break;
            case 5:
                display(root);
                break;
            case 6:
                exit(0);


        }
```


D:\chethan\DIP\BST.exe
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to insert: 23
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to insert: 11
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to insert: 455
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 5
BST Elements (Inorder): 11 23 455
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 3
Preorder Traversal: 23 11 455
--- Binary Search Tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 4
postorder Traversal:11 455 23
--- Binary Search Tree Menu ---
1. Insert into BST
Search
ENG
IN
08:49
08-12-2025

DATE:

8) Binary search tree implement and traverse the tree all the methods inorder, preorder, postorder.

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct Node* left; right;
};
struct Node * createNode (int value){
    struct Node *newNode = (struct Node *)malloc (sizeof (struct
    newNode -> data = value;
    newNode -> left = newNode -> right = NULL;
    return newNode;
}
Struct Node * insert (struct Node* root, int value)
{
    if (root == NULL)
        return createNode(value);
    if( value < root -> data)
        root -> left = insert (root -> left, value);
    elseif ( value > root -> data)
        root -> right = insert (root -> right, value);
    return root;
}
void inorder (struct Node * root) {
    if(root == NULL) return;
    inorder (root -> left);
    printf ("%d", root -> data);
    inorder (root -> right);
}
```

```
void preorder(struct Node* root) {
    if(root == NULL) return;
    printf("%d", root->data);
    preorder(root->left);
    preorder(root->right);
}
void postorder(struct Node* root) {
    if(root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d", root->data);
}
void display(struct node* root) {
    printf("BST Elements (Inorder): ");
    inorder(root);
    printf("\n");
}
int main() {
    struct Node* root = NULL;
    int choice, value;
    while(1) {
        printf("\n--- Binary search Tree Menu ---\n");
        printf("1. Insert into BST\n");
        printf("2. Inorder Traversal\n");
        printf("3. preorder Traversal\n");
        printf("4. postorder Traversal\n");
        printf("5. display BST\n");
        printf("6 Exit\n");
        printf("Enter choice:");
        scanf("%d", &choice);
```

```
switch(choice) {
    case 1:
        printf("Enter value of insert);
        scanf("%d", &value);
        root = insert (root, value);
        break;
    case 2:
        printf("Inorder Traversal:");
        inorder (root);
        printf("\n");
        break;
    case 3:
        printf(" preorder traversal");
        preorder(root);
        printf("\n");
        break;
    case 4:
        printf("postorder Traversal:");
        postorder (root);
        printf("\n");
        break;
    case 5:
        display (root);
        break;
    case 6:
        Exit(0);
    }
}
}
```

DATE:

Output

... Binary search tree Menu ...
1. Insert into BST
2. Inorder Traversal
3. preorder Traversal
4. postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to insert: 23 45 567

Enter choice: 2
Inorder Traversal: 23 45 567
Enter choice: 3
preorder Traversal: 23 45 567
Enter choice: 4
postorder Traversal: 567 45 23
Enter choice: 5
BST (element): 23 45 567
Enter choice: 6
process returned 0 (0x0) execution time: 343.927s