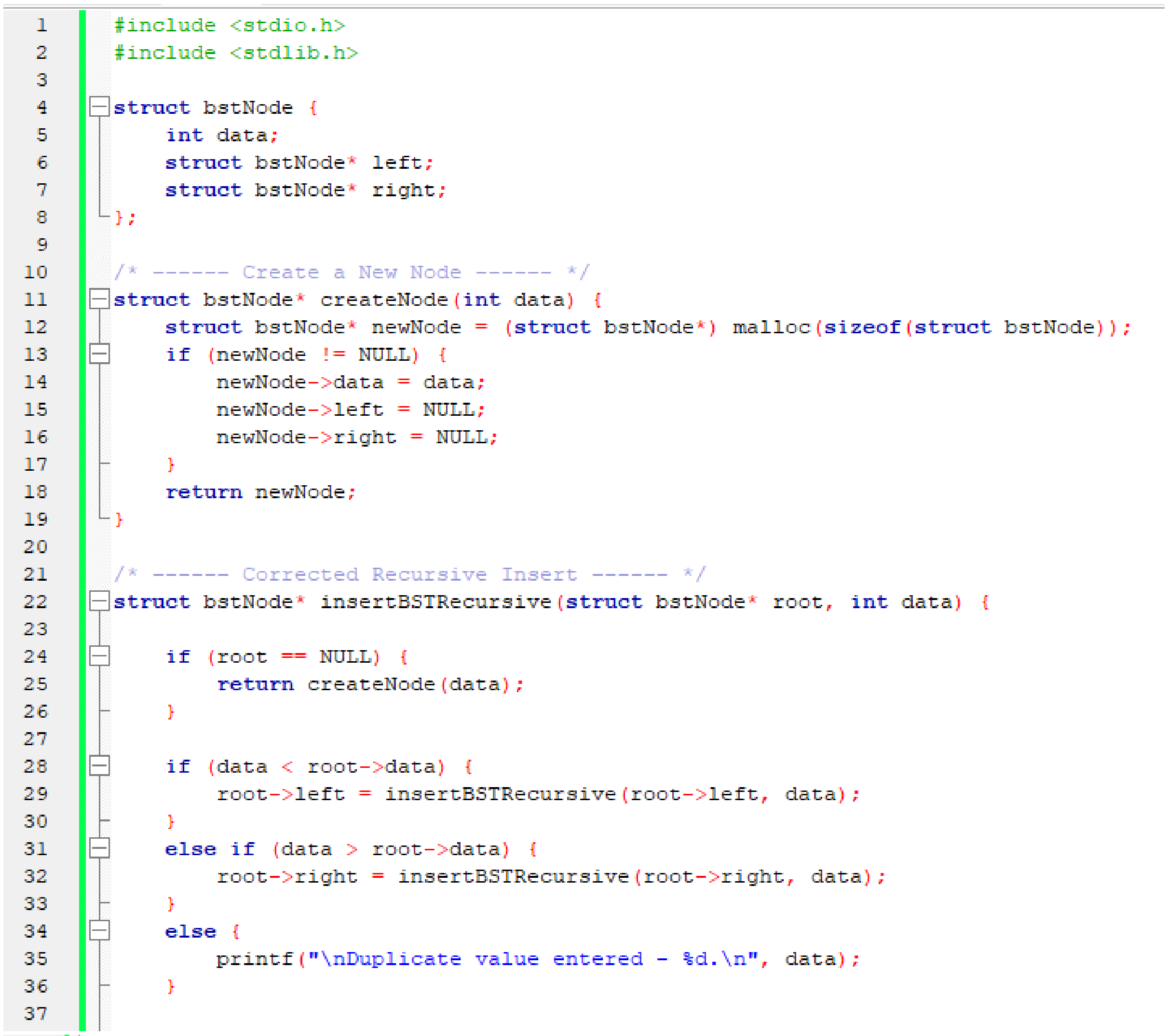
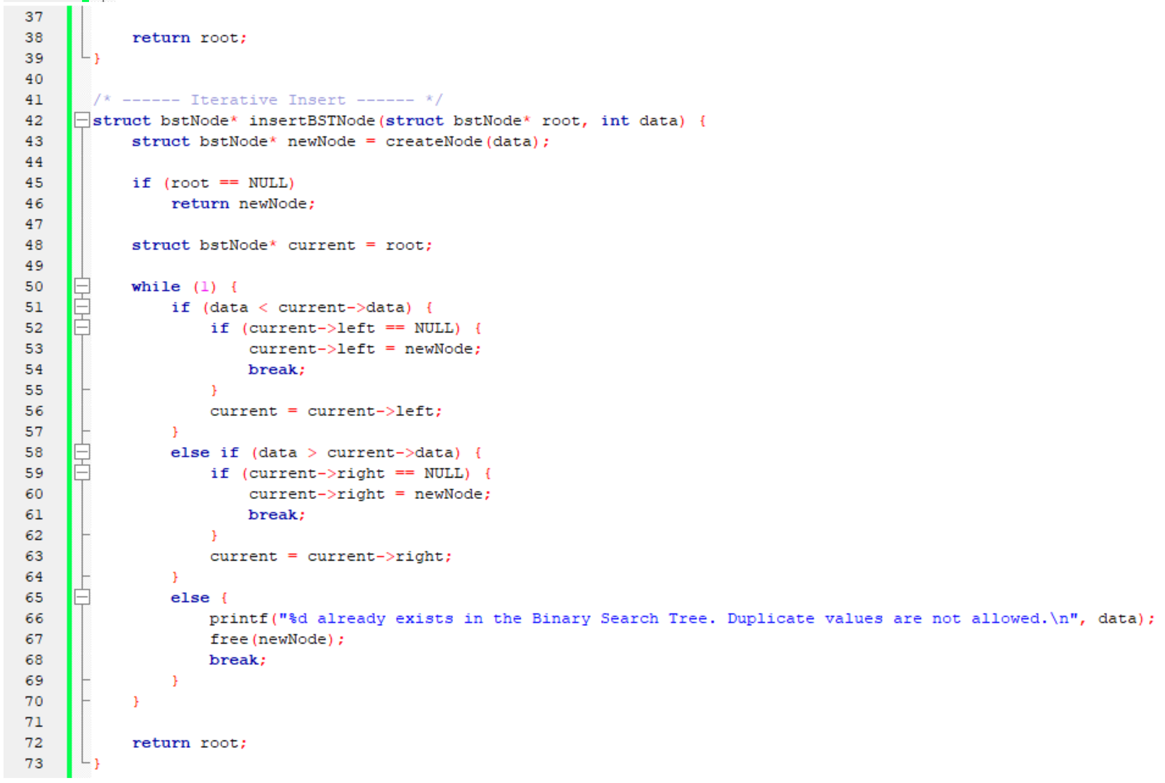
BINARY SEARCH TREE: -

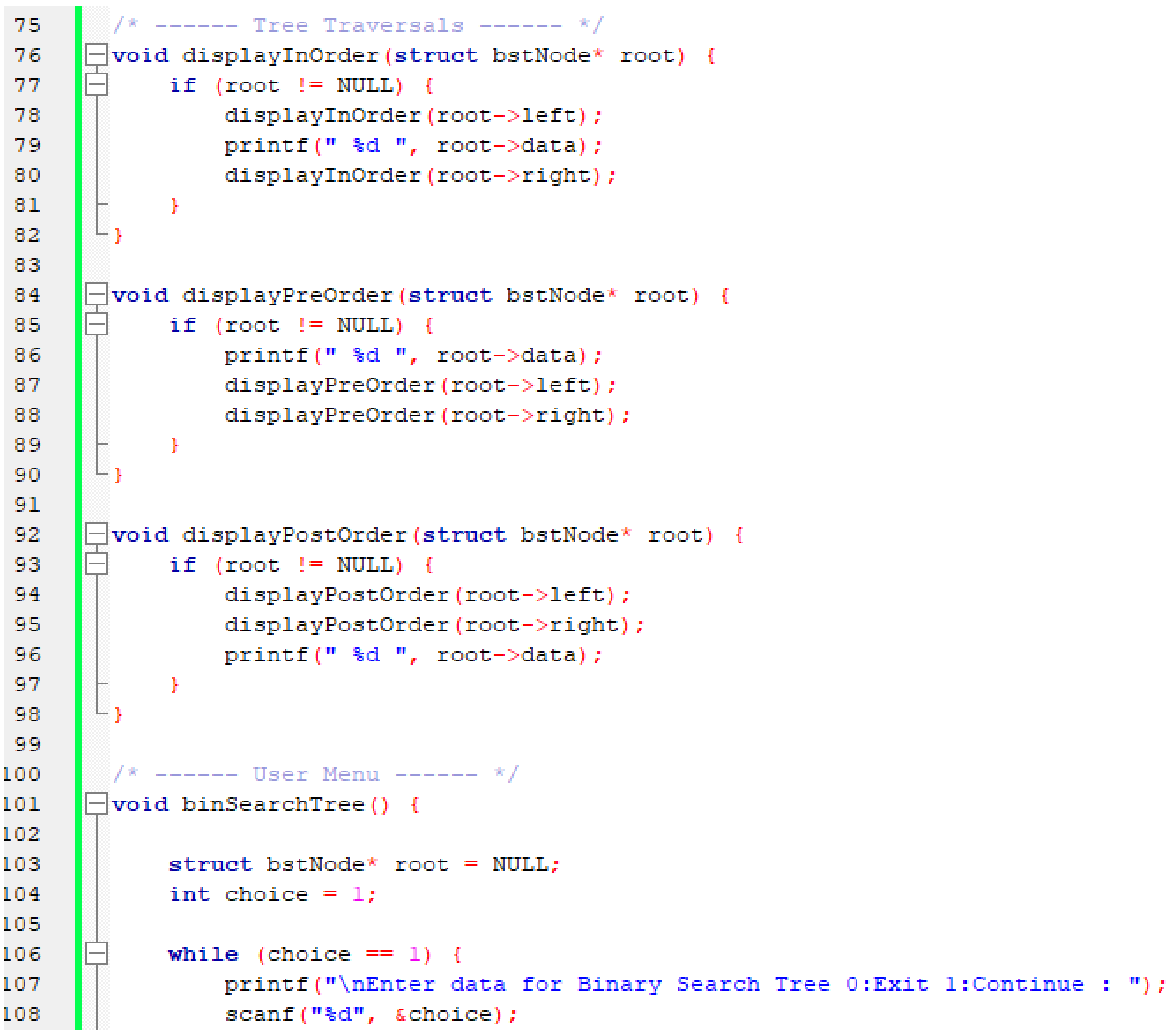
CODE: -



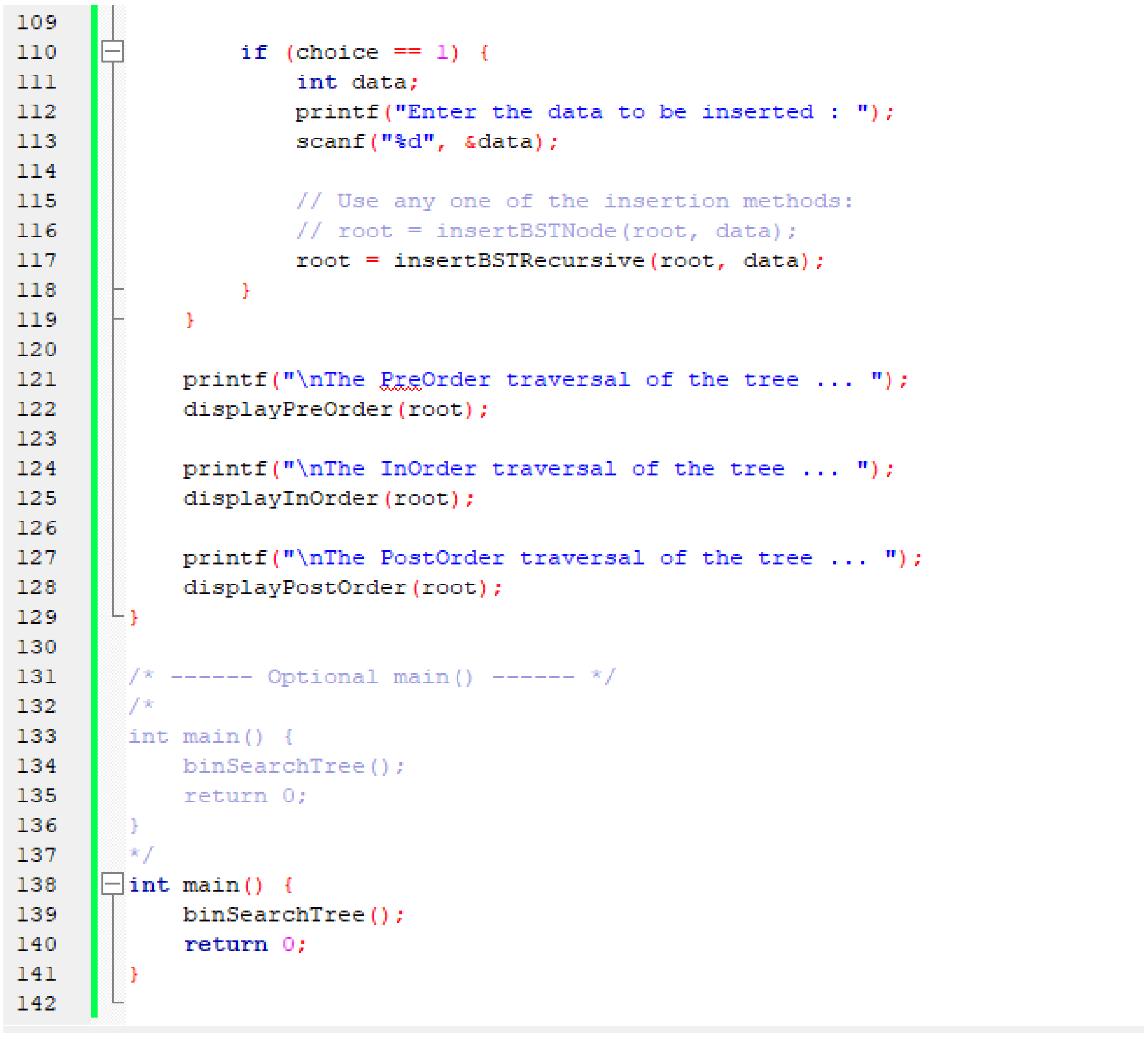
```c
#include <stdio.h>
#include <stdlib.h>

struct bstNode {
    int data;
    struct bstNode* left;
    struct bstNode* right;
};

/* ------ Create a New Node ------ */
struct bstNode* createNode(int data) {
    struct bstNode* newNode = (struct bstNode*) malloc(sizeof(struct bstNode));
    if (newNode != NULL) {
        newNode->data = data;
        newNode->left = NULL;
        newNode->right = NULL;
    }
    return newNode;
}

/* ------ Corrected Recursive Insert ------ */
struct bstNode* insertBSTRecursive(struct bstNode* root, int data) {

    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insertBSTRecursive(root->left, data);
    }
    else if (data > root->data) {
        root->right = insertBSTRecursive(root->right, data);
    }
    else {
        printf("\nDuplicate value entered - %d.\n", data);
    }

```



```c

    return root;
}

/* ------ Iterative Insert ------ */
struct bstNode* insertBSTNode(struct bstNode* root, int data) {
    struct bstNode* newNode = createNode(data);

    if (root == NULL)
        return newNode;

    struct bstNode* current = root;

    while (1) {
        if (data < current->data) {
            if (current->left == NULL) {
                current->left = newNode;
                break;
            }
            current = current->left;
        }
        else if (data > current->data) {
            if (current->right == NULL) {
                current->right = newNode;
                break;
            }
            current = current->right;
        }
        else {
            printf("%d already exists in the Binary Search Tree. Duplicate values are not allowed.\n", data);
            free(newNode);
            break;
        }
    }

    return root;
}
```

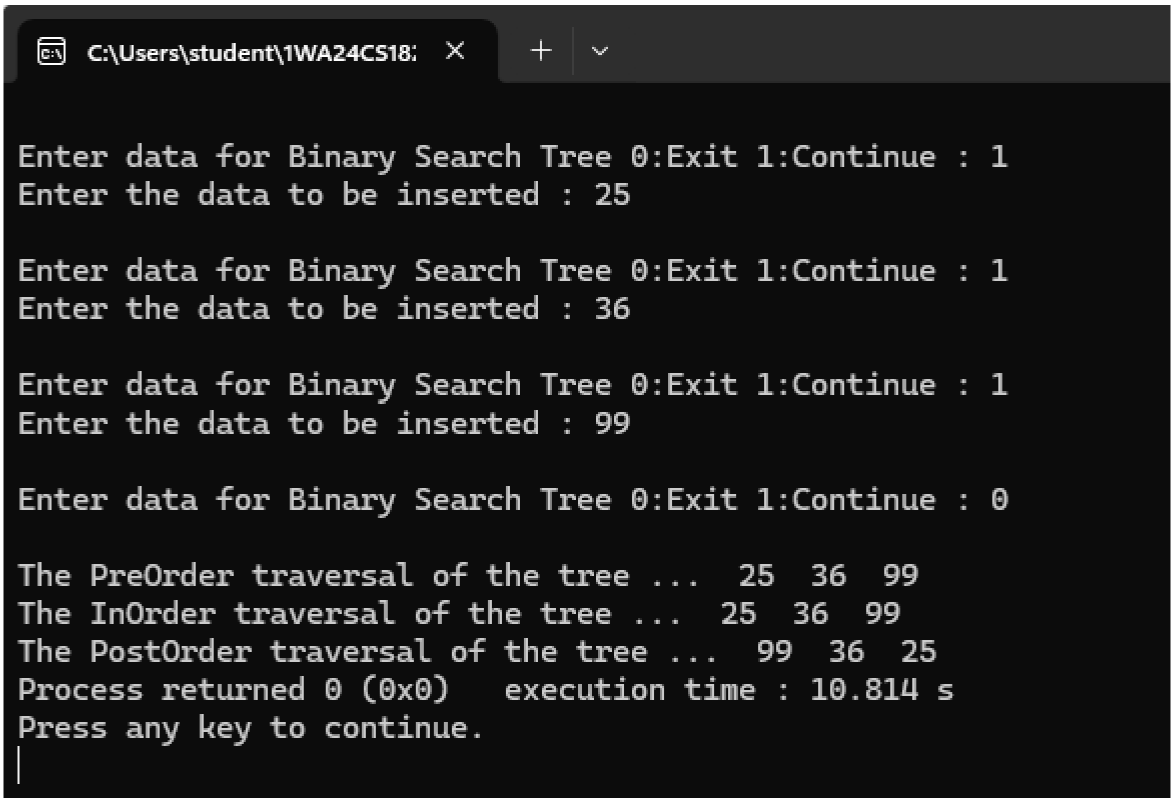


```c
/* ------ Tree Traversals ------ */
void displayInOrder(struct bstNode* root) {
    if (root != NULL) {
        displayInOrder(root->left);
        printf(" %d ", root->data);
        displayInOrder(root->right);
    }
}

void displayPreOrder(struct bstNode* root) {
    if (root != NULL) {
        printf(" %d ", root->data);
        displayPreOrder(root->left);
        displayPreOrder(root->right);
    }
}

void displayPostOrder(struct bstNode* root) {
    if (root != NULL) {
        displayPostOrder(root->left);
        displayPostOrder(root->right);
        printf(" %d ", root->data);
    }
}

/* ------ User Menu ------ */
void binSearchTree() {

    struct bstNode* root = NULL;
    int choice = 1;

    while (choice == 1) {
        printf("\nEnter data for Binary Search Tree 0:Exit 1:Continue : ");
        scanf("%d", &choice);
```



```c

        if (choice == 1) {
            int data;
            printf("Enter the data to be inserted : ");
            scanf("%d", &data);

            // Use any one of the insertion methods:
            // root = insertBSTNode(root, data);
            root = insertBSTRecursive(root, data);
        }
    }

    printf("\nThe PreOrder traversal of the tree ... ");
    displayPreOrder(root);

    printf("\nThe InOrder traversal of the tree ... ");
    displayInOrder(root);

    printf("\nThe PostOrder traversal of the tree ... ");
    displayPostOrder(root);
}

/* ------ Optional main() ------ */
/*
int main() {
    binSearchTree();
    return 0;
}
*/
int main() {
    binSearchTree();
    return 0;
}

```

OUTPUT: -



```
Enter data for Binary Search Tree 0:Exit 1:Continue : 1
Enter the data to be inserted : 25

Enter data for Binary Search Tree 0:Exit 1:Continue : 1
Enter the data to be inserted : 36

Enter data for Binary Search Tree 0:Exit 1:Continue : 1
Enter the data to be inserted : 99

Enter data for Binary Search Tree 0:Exit 1:Continue : 0

The PreOrder traversal of the tree ...  25  36  99
The InOrder traversal of the tree ...  25  36  99
The PostOrder traversal of the tree ...  99  36  25
Process returned 0 (0x0)   execution time : 10.814 s
Press any key to continue.
```