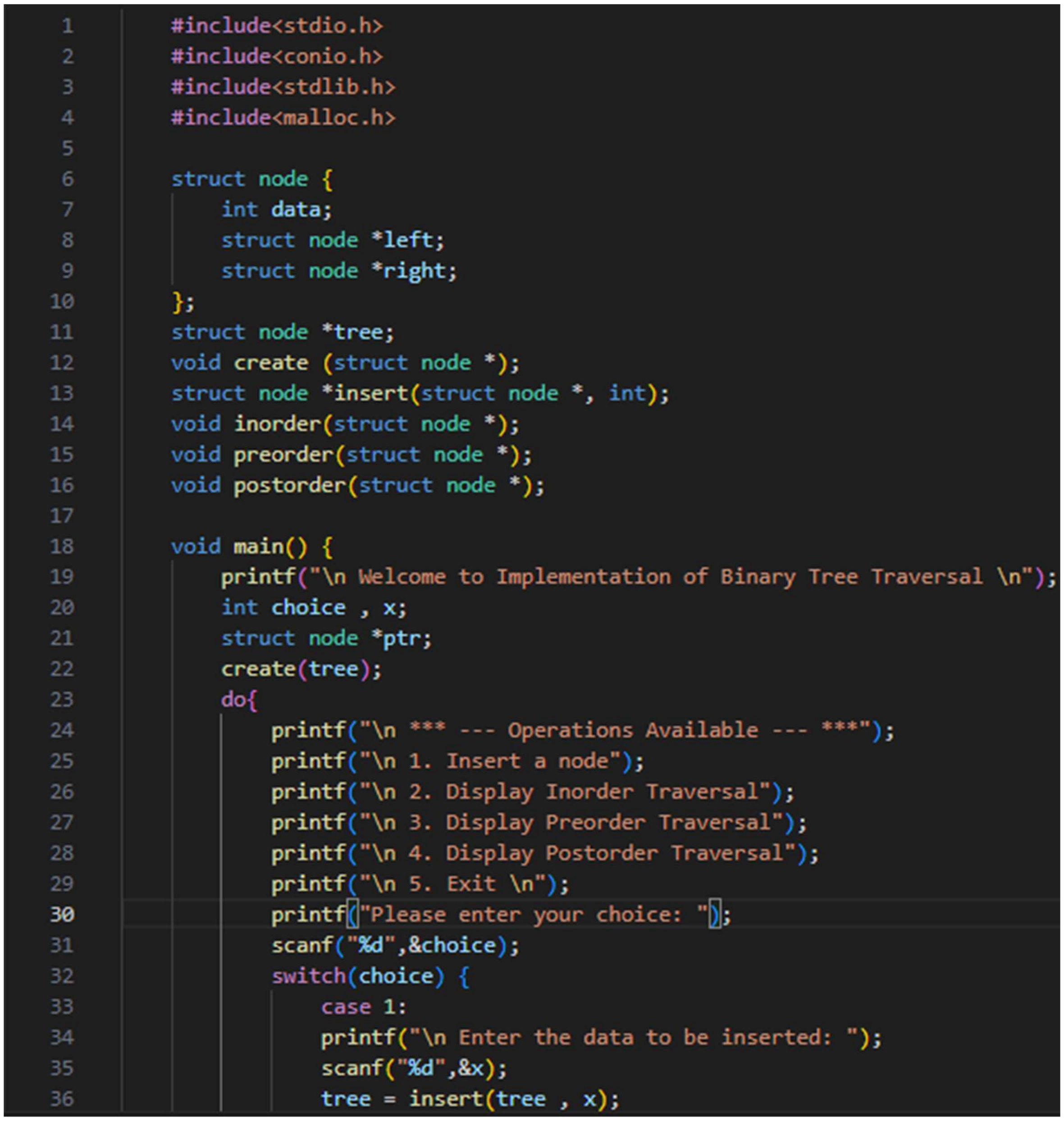
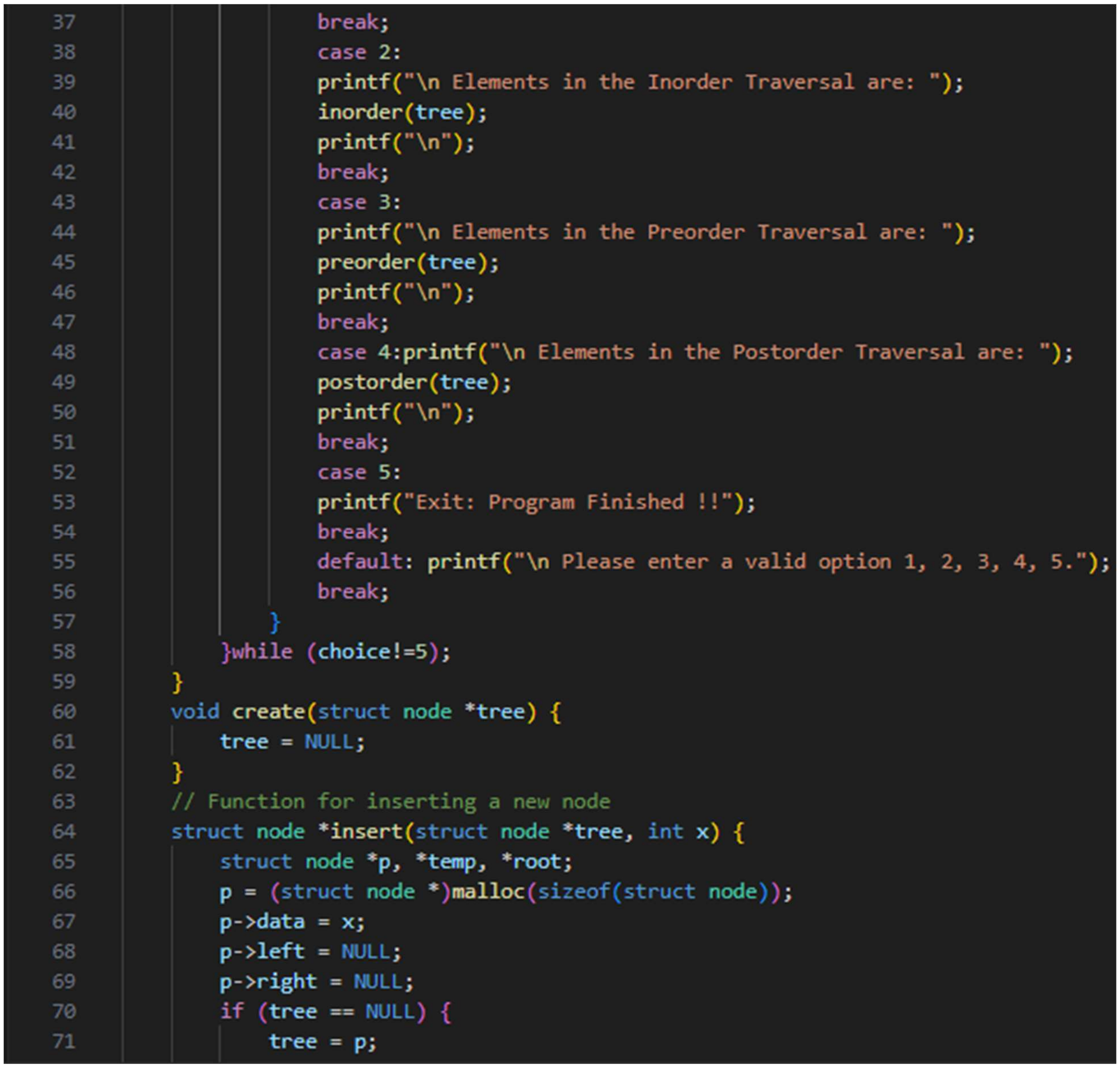
Experiment No. 6

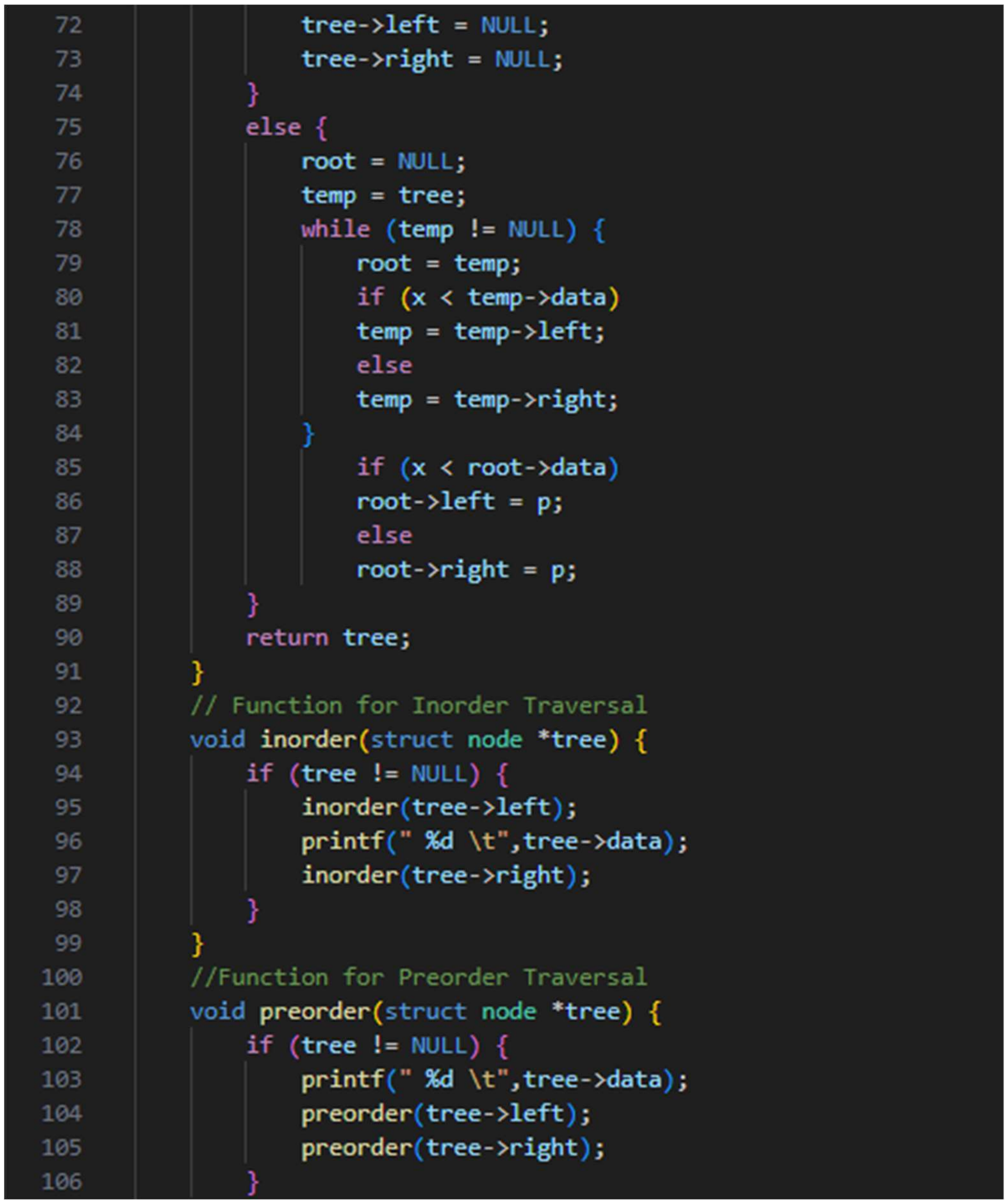
Program:



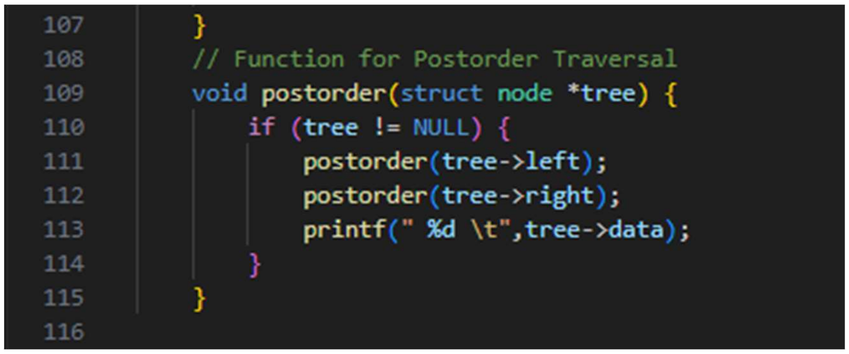
```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>

struct node {
    int data;
    struct node *left;
    struct node *right;
};
struct node *tree;
void create (struct node *);
struct node *insert(struct node *, int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);

void main() {
    printf("\n Welcome to Implementation of Binary Tree Traversal \n");
    int choice , x;
    struct node *ptr;
    create(tree);
    do{
        printf("\n *** --- Operations Available --- ***");
        printf("\n 1. Insert a node");
        printf("\n 2. Display Inorder Traversal");
        printf("\n 3. Display Preorder Traversal");
        printf("\n 4. Display Postorder Traversal");
        printf("\n 5. Exit \n");
        printf("Please enter your choice: ");
        scanf("%d",&choice);
        switch(choice) {
            case 1:
            printf("\n Enter the data to be inserted: ");
            scanf("%d",&x);
            tree = insert(tree , x);
```



```c
                break;
                case 2:
                printf("\n Elements in the Inorder Traversal are: ");
                inorder(tree);
                printf("\n");
                break;
                case 3:
                printf("\n Elements in the Preorder Traversal are: ");
                preorder(tree);
                printf("\n");
                break;
                case 4:printf("\n Elements in the Postorder Traversal are: ");
                postorder(tree);
                printf("\n");
                break;
                case 5:
                printf("Exit: Program Finished !!");
                break;
                default: printf("\n Please enter a valid option 1, 2, 3, 4, 5.");
                break;
            }
        }while (choice!=5);
    }
    void create(struct node *tree) {
        tree = NULL;
    }
    // Function for inserting a new node
    struct node *insert(struct node *tree, int x) {
        struct node *p, *temp, *root;
        p = (struct node *)malloc(sizeof(struct node));
        p->data = x;
        p->left = NULL;
        p->right = NULL;
        if (tree == NULL) {
            tree = p;
```

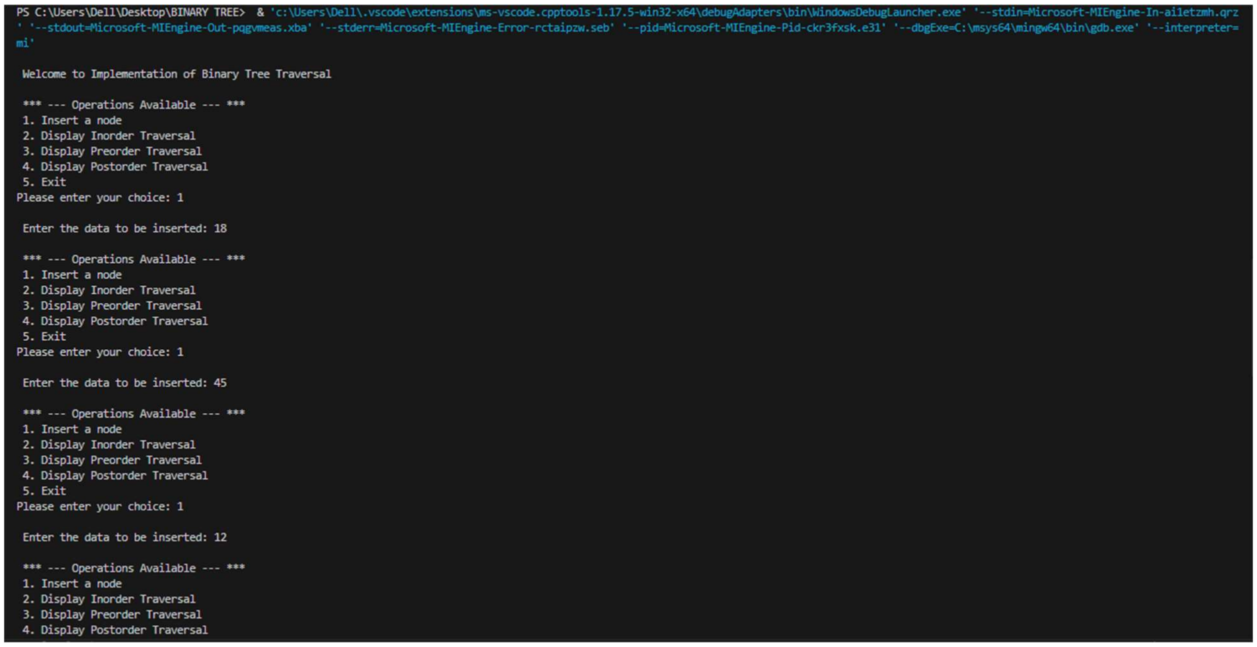


```c
        tree->left = NULL;
        tree->right = NULL;
    }
    else {
        root = NULL;
        temp = tree;
        while (temp != NULL) {
            root = temp;
            if (x < temp->data)
            temp = temp->left;
            else
            temp = temp->right;
        }
            if (x < root->data)
            root->left = p;
            else
            root->right = p;
    }
    return tree;
}
// Function for Inorder Traversal
void inorder(struct node *tree) {
    if (tree != NULL) {
        inorder(tree->left);
        printf(" %d \t",tree->data);
        inorder(tree->right);
    }
}
//Function for Preorder Traversal
void preorder(struct node *tree) {
    if (tree != NULL) {
        printf(" %d \t",tree->data);
        preorder(tree->left);
        preorder(tree->right);
    }
```

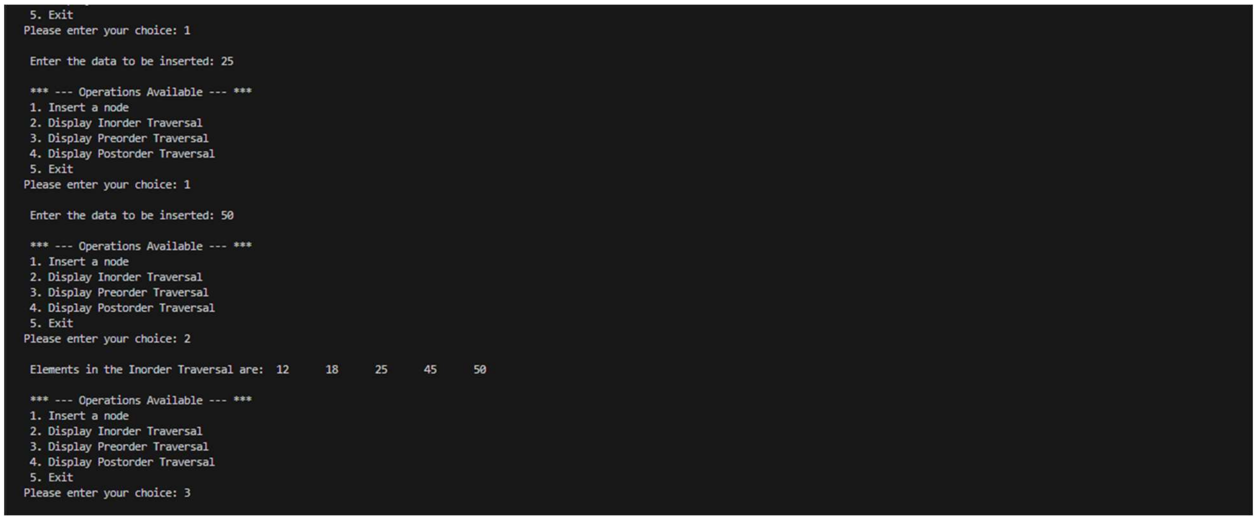


```c
}
// Function for Postorder Traversal
void postorder(struct node *tree) {
    if (tree != NULL) {
        postorder(tree->left);
        postorder(tree->right);
        printf(" %d \t",tree->data);
    }
}

```

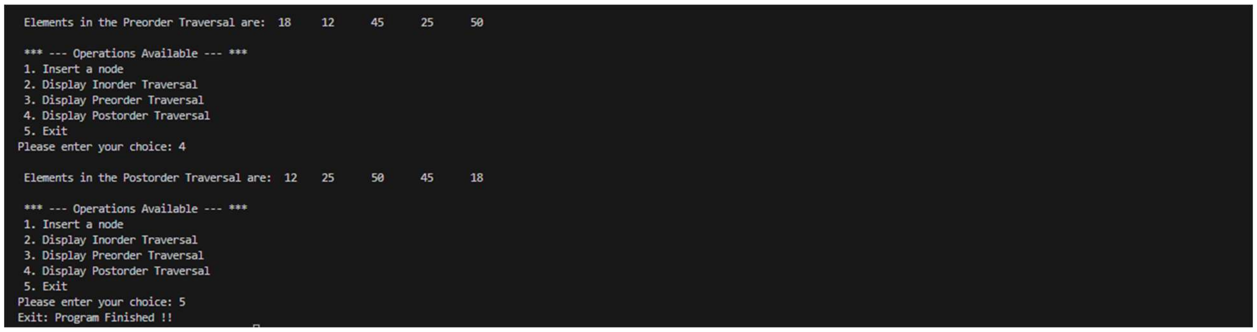
Output:



```
PS C:\Users\Dell\Desktop\BINARY TREE>  & 'c:\Users\Dell\.vscode\extensions\ms-vscode.cpptools-1.17.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ai1etzmh.qrz
' '--stdout=Microsoft-MIEngine-Out-pqgvmeas.xba' '--stderr=Microsoft-MIEngine-Error-rctaipzw.seb' '--pid=Microsoft-MIEngine-Pid-ckr3fxsk.e31' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=
mi'

 Welcome to Implementation of Binary Tree Traversal

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 1

 Enter the data to be inserted: 18

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 1

 Enter the data to be inserted: 45

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 1

 Enter the data to be inserted: 12

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
```



```
 5. Exit
Please enter your choice: 1

 Enter the data to be inserted: 25

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 1

 Enter the data to be inserted: 50

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 2

 Elements in the Inorder Traversal are:  12      18      25      45      50

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 3
```



```
 Elements in the Preorder Traversal are:  18      12      45      25      50

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 4

 Elements in the Postorder Traversal are:  12     25      50      45      18

 *** --- Operations Available --- ***
 1. Insert a node
 2. Display Inorder Traversal
 3. Display Preorder Traversal
 4. Display Postorder Traversal
 5. Exit
Please enter your choice: 5
Exit: Program Finished !!
```