Binary Search Tree

Q Writea a program to construct a binary search Tree, and to traverse the tree using all methods, i.e, In-order, pre-order and post order

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

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

typedef struct Node node;

node * createNode( int data){
            node * new1 = (node *) malloc( sizeof (node));
            new1 -> data = data;
            new1 -> left = NULL;
            new1 -> right = NULL;
            return new1;

node * insertNode (node * root, int data){
        if (root == NULL){
            return createNode (data);
        }
        if (data < root -> data){
            root -> left = insertNode (root -> left, data);
        } else{
            root -> right = insertNode (root -> right, data);
        }
        return root;
```

```
void inOrderTraversal (node * root){
        if( root != NULL){
                inorderTraversal (root -> left);
                printf("%d", root -> data);
                inorderTraversal (root -> right);
        }
}

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

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

void main (){
        node * root = NULL;
        int choice, value;
        while (1){
                printf ("\n1. Insert \n2. In-Order Traversal \n3. Pre-Order
                Traversal \n 4. Post-Order Traversal \n5. Exit \n");
                printf("Enter Your Choice: ");
                scanf("%d", &choice);
switch (choice) {
    case 1:
        printf("Enter Value to Insert: ");
        scanf("%d", & value);
        root = insertNode (root, value);
        break;
    case 2:
        printf(" In-Order Traversal");
        inorder Traversal (root):
        break;
    case 3:
        printf("Pre-Order Traversal");
        preorder Traversal (root);
        break;
    case 4:
        printf(" Post-Order Traversal");
        postorder Traversal (root);
        break;
    case 5:
        exit(0);
    }
  }
}
```

Output

```
1. Insert
2. In-Order Traversal
3. Pre-Order Traversal
4. Post-Order Traversal
5. Exit
→ Enter Your Choice: 1
  Enter Value to Insert: 50
```

```
→ Enter Your Choice: 1
  Enter Value to Insert: 40

→ Enter Your Choice : 1
  Enter Value to Insert : 75

→ Enter Your Choice: 1
  Enter Value to Insert : 10

→ Enter Your Choice: 1
  Enter Value to Insert: 25

→ Enter Your Choice: 1
  Enter Value to Insert: 80

→ Enter Your Choice: 1
  Enter Value to Insert: 20

→ Enter Your Choice: 2
  In-Order Traversal: 10  20  25  40  50  75  80

→ Enter Your Choice: 3
  Pre-Order Traversal: 50 40 10 25 20 75 80

→ Enter Your Choice: 4
  Post-Order Traversal: 20  25  10  40  80  75  50

→ Enter Your Choice: 5
  Tree Representation:
                                    80
                    75
          50
                    40
                                    25
                                              20
```