# Exp 6

**Name : Yash Bid**
**Roll no : 05**

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

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

struct node *tree = NULL; // Initialize the tree

void create(struct node **);
struct node *insert(struct node *, int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);

int main() {
    printf("\n--- Welcome To Implementation Of Binary Tree Traversals ---\n");
    int choice, x;
    struct node *ptr;

    create(&tree);

    do {
        printf("\n*** --- Operations Available --- ***\n");
        printf("1. Insert a Node\n");
        printf("2. Display Inorder Traversal\n");
        printf("3. Display Preorder Traversal\n");
        printf("4. Display Postorder Traversal\n");
        printf("5. Exit\n");
        printf("Please enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
```

```
            printf("Enter the data to be inserted: ");
            scanf("%d", &x);
            tree = insert(tree, x);
            break;
        case 2:
            printf("Elements in the inorder traversal are: ");
            inorder(tree);
            printf("\n");
            break;
        case 3:
            printf("Elements in the preorder traversal are: ");
            preorder(tree);
            printf("\n");
            break;
        case 4:
            printf("Elements in the postorder traversal are: ");
            postorder(tree);
            printf("\n");
            break;
        case 5:
            printf("Exit: Program Finished !!\n");
            break;
        default:
            printf("Please enter a valid option (1, 2, 3, 4, 5).\n");
            break;
      }
   } while (choice != 5);

   return 0;
}

void create(struct node **tree) {
   *tree = NULL;
}

struct node *insert(struct node *tree, int x) {
   struct node *p;
   if (tree == NULL) {
      p = (struct node *)malloc(sizeof(struct node));
      p->data = x;
      p->left = NULL;
      p->right = NULL;
      return p;
   }
```

```
    if (x < tree->data) {
        tree->left = insert(tree->left, x);
    } else {
        tree->right = insert(tree->right, x);
    }
    return tree;
}

void inorder(struct node *tree) {
    if (tree != NULL) {
        inorder(tree->left);
        printf("%d \t", tree->data);
        inorder(tree->right);
    }
}

void preorder(struct node *tree) {
    if (tree != NULL) {
        printf("%d \t", tree->data);
        preorder(tree->left);
        preorder(tree->right);
    }
}

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

```
*** --- 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: 75

*** --- 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: 10       71      72      73      75

*** --- 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: 10      71      72      73      75

*** --- 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: 75     73      72      71      10
```

```
*** ... 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 !!
dl0410@itadmin:~/Desktop/temp$
```