DAY 9 LAB

1.BINARY TREE

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

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

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

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

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

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

int main() {
    struct Node* root = createNode(1);
    root->left = createNode(2);
    root->right = createNode(3);
    root->left->left = createNode(4);
    root->left->right = createNode(5);

    printf("In-order traversal: ");
    inOrder(root);
    printf("\n");

    printf("Pre-order traversal: ");
    preOrder(root);
    printf("\n");

    printf("Post-order traversal: ");
    postOrder(root);
    printf("\n");
    return 0;
}
```

OUTPUT:

```
In-order traversal: 4 2 5 1 3
Pre-order traversal: 1 2 4 5 3
Post-order traversal: 4 5 2 3 1
=== Code Execution Successful ===
```

2. write a C program for binary search tree

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* left;
    struct node* right;
};
struct node* createNode(int value) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
struct node* insert(struct node* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }
    return root;
}
void inorderTraversal(struct node* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
int main() {
    struct node* root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);
    printf("Inorder traversal of the binary search tree: ");
    inorderTraversal(root);
    return 0;
}
```

OUTPUT:

```
Inorder traversal of the binary search tree: 20 30 40 50 60 70 80
```