DAY-9-DS

1. #include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* left;

struct Node\* right;

} Node;

Node\* createNode(int value) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = value;

newNode->left = NULL;

newNode->right = NULL;

return newNode;

}

Node\* insert(Node\* root, int value) {

if (root == NULL) return createNode(value);

if (value < root->data)

root->left = insert(root->left, value);

else if (value > root->data)

root->right = insert(root->right, value);

return root;

}

void inorderTraversal(Node\* root) {

if (root != NULL) {

inorderTraversal(root->left);

printf("%d ", root->data);

inorderTraversal(root->right);

}

}

void destroyTree(Node\* root) {

if (root != NULL) {

destroyTree(root->left);

destroyTree(root->right);

free(root);

}

}

int main() {

Node\* root = NULL;

root = insert(root, 50);

insert(root, 30);

insert(root, 70);

insert(root, 20);

insert(root, 40);

insert(root, 60);

insert(root, 80);

printf("In-order traversal: ");

inorderTraversal(root);

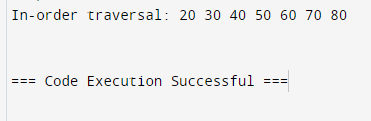
printf("\n");

destroyTree(root);

return 0;

}

Output:



2. #include <stdio.h>

#include <stdlib.h>

typedef struct Node {

int data;

struct Node\* left;

struct Node\* right;

} Node;

Node\* createNode(int value) {

Node\* newNode = (Node\*)malloc(sizeof(Node));

newNode->data = value;

newNode->left = newNode->right = NULL;

return newNode;

}

Node\* insert(Node\* root, int value) {

if (root == NULL) return createNode(value);

if (value < root->data) root->left = insert(root->left, value);

else if (value > root->data) root->right = insert(root->right, value);

return root;

}

Node\* findMin(Node\* root) {

while (root->left != NULL) root = root->left;

return root;

}

Node\* delete(Node\* root, int value) {

if (root == NULL) return root;

if (value < root->data) root->left = delete(root->left, value);

else if (value > root->data) root->right = delete(root->right, value);

else {

if (root->left == NULL) {

Node\* temp = root->right;

free(root);

return temp;

} else if (root->right == NULL) {

Node\* temp = root->left;

free(root);

return temp;

}

Node\* temp = findMin(root->right);

root->data = temp->data;

root->right = delete(root->right, temp->data);

}

return root;

}

void inorderTraversal(Node\* root) {

if (root != NULL) {

inorderTraversal(root->left);

printf("%d ", root->data);

inorderTraversal(root->right);

}

}

int main() {

Node\* root = NULL;

root = insert(root, 50);

insert(root, 30); insert(root, 70);

insert(root, 20); insert(root, 40);

insert(root, 60); insert(root, 80);

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

root = delete(root, 20);

printf("After deletion of 20: "); inorderTraversal(root); printf("\n");

return 0;

}

Output:

