***// BINARY SEARCH TREE***

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

typedef struct node

{

int data;

struct node \*left;

struct node \*right;

} Node;

typedef struct tree

{

Node \*root;

} Tree;

void tree\_initialize(Tree \*tree);

void tree\_insert(Tree \*tree, int data);

void inorder(Tree \*tree);

void preorder(Tree \*tree);

void postorder(Tree \*tree);

void tree\_destroy(Tree \*tree);

int main()

{

int choice, loop = 1;

Tree my\_tree;

tree\_initialize(&my\_tree);

while (loop)

{

scanf("%d", &choice);

switch (choice)

{

int number\_of\_elements, element, index;

int data;

case 1:

*/\* Insert elements \*/*

scanf("%d", &element);

tree\_insert(&my\_tree, element);

break;

case 2:

*/\* Print elements in the inorder fashion \*/*

preorder(&my\_tree);

break;

case 3:

*/\* Print elements in the preorder fashion \*/*

inorder(&my\_tree);

break;

case 4:

*/\* Print elements in the postorder fashion \*/*

postorder(&my\_tree);

break;

default:

tree\_destroy(&my\_tree);

loop = 0;

break;

}

}

return 0;

}

void tree\_initialize(Tree \*tree)

{

tree->root==NULL;

}

Node\* create\_Node(int data){

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

new->data = data;

new->left = new->right = NULL;

return new;

}

Node\* rec\_Insert\_Node(Node \*root, int data){

Node \*new = create\_Node(data);

if (root == NULL){

root = new;

}

else if (root->data > new->data){

root->left = rec\_Insert\_Node(root->left, data);

}

else if (root->data < new->data){

root->right = rec\_Insert\_Node(root->right, data);

}

return root;

}

void tree\_insert(Tree \*tree, int data){

tree->root=rec\_Insert\_Node(tree->root, data);

}

void tree\_inorder(Node \*r)

{

if(r!=NULL){

tree\_inorder(r->left);

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

tree\_inorder(r->right);

}

}

void tree\_preorder(Node \*r)

{

if (r != NULL){

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

tree\_preorder(r->left);

tree\_preorder(r->right);

}

}

void tree\_postorder(Node \*r)

{

if (r != NULL){

tree\_postorder(r->left);

tree\_postorder(r->right);

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

}

}

void postorder(Tree \*t)

{

tree\_postorder(t->root);

printf("\n");

}

void preorder(Tree \*t)

{

tree\_preorder(t->root);

printf("\n");

}

void inorder(Tree \*t)

{

tree\_inorder(t->root);

printf("\n");

}

void destroy(Node \*r)

{

r=NULL;

}

void tree\_destroy(Tree \*t)

{

destroy(t->root);

}