**binary-tree.c**

//write a propgram to implement binary tree and show the all order traversal of it.(pre-order, in-order, post-order)

#include <stdio.h>

#include <stdlib.h>

typedef struct Binary\_Tree{

int data;

struct Binary\_Tree \*l;

struct Binary\_Tree \*r;

}btree;

void add(btree \*\*t, int new\_data){

btree \*new\_btree = (btree\*)malloc(sizeof(btree));

new\_btree->data = new\_data;

new\_btree->l = NULL;

new\_btree->r = NULL;

\*t = new\_btree;

}

void display(btree \*t){

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

}

void preorderTraversal(btree\* root) {

if (root) {

display(root);

preorderTraversal(root->l);

preorderTraversal(root->r);

}

}

void inorderTraversal(btree\* root) {

if (root) {

preorderTraversal(root->l);

display(root);

preorderTraversal(root->r);

}

}

void postorderTraversal(btree\* root) {

if (root) {

preorderTraversal(root->l);

preorderTraversal(root->r);

display(root);

}

}

void main(){

btree \*t1 = NULL;

add(&t1,1);

add(&t1->l,2);

add(&t1->r,3);

add(&t1->l->l,4);

add(&t1->r->l,5);

add(&t1->l->r,6);

add(&t1->r->r,7);

add(&t1->l->l->r,8);

add(&t1->r->r->l,9);

printf("PreOrder: ");

preorderTraversal(t1);

printf("\nInorder: ");

inorderTraversal(t1);

printf("\nPostOrder: ");

postorderTraversal(t1);

}

**OUTPUT**

PS S:\WorkSpace\CollegeWork\DataStructure\Temp> gcc .\binary-tree.c

PS S:\WorkSpace\CollegeWork\DataStructure\Temp> ./a

PreOrder: 1 2 4 8 6 3 5 7 9

Inorder: 2 4 8 6 1 3 5 7 9

PostOrder: 2 4 8 6 3 5 7 9 1

PS S:\WorkSpace\CollegeWork\DataStructure\Temp>