Problem Statement

Write a Program to implement following operations on Binary Tree.

- ➢ Create a Binary Tree
- ➢ Traverse a Tree in in-order
- ➢ Traverse a tree in post-order
- ➢ Traverse a tree in pre-order

```
////// Include files //////////////
#include<iostream>
#include<stdlib.h>
using namespace std;
///// Create structure to represent node of a tree /////
struct Tree {
        char data;       // data of the node
        Tree *left; // left pointer of node
        Tree *right;// right pointer of node
};
typedef struct Tree tree;
///// declare functions ///////////////
tree *root;
void insert(char x, char *dir);
tree* create_node(char x);
void remove();
void traverse_preorder(tree *t);
void traverse_postorder(tree *t);
void traverse_inorder(tree *t);
//////// implement main ////////////////////////////
int main() {
        /*
        root = create_node('A');
        insert('B', "0");                    //0 for left child and 1 for right child
        insert('C', "1");
        insert('D', "00");
        insert('E', "01");
        insert('F', "11");
        insert('G', "110");
        */
        root = create_node('+');
        insert('*',"0");
        insert('E',"1");
        insert('*',"00");
        insert('D',"01");
        insert('/',"000");
        insert('C',"001");
        insert('A',"0000");
```

```
	insert('B',"0001");
	cout<<endl<<"Preorder Traversal: "<<endl;
	traverse_preorder(root);
	cout<<endl<<"Postorder Traversal: "<<endl;
	traverse_postorder(root);
	cout<<endl<<"Inorder Traversal: "<<endl;
	traverse_inorder(root);
	return 0;
}
////// it will create root node ///
tree* create_node(char x) {
	tree *newnode = (tree*)malloc(sizeof(tree));
	newnode->data = x;
	newnode->left = NULL;
	newnode->right = NULL;
	return newnode;
}
//////// it will insert new node to the tree /////
void insert(char x, char *dir) { //dir for direction
	tree *t,*t1;
	tree *newnode = create_node(x);
	t = root;
	int i = 0;
	while(t) {
		while(dir[i]!=NULL) {
			if(dir[i]=='0') {
			t1 = t;
			t = t->left;
			}

			else {
			t1 = t;
			t = t->right;
			}
			i++;
		}
	}
	if(dir[--i]=='0')
	t1->left = newnode;
	else
	t1->right = newnode;
}
///// it will trverse the tree in preorder /////
void traverse_preorder(tree *t) {
	if(t) {
		cout<<t->data<<", ";
		traverse_preorder(t->left);
		traverse_preorder(t->right);
```

```
	}

}
///// it will trverse the tree in postorder /////
void traverse_postorder(tree *t) {
	if(t) {

		traverse_postorder(t->left);
		traverse_postorder(t->right);
		cout<<t->data<<", ";
	}

}
///// it will trverse the tree in inorder /////
void traverse_inorder(tree *t) {
	if(t) {

		traverse_inorder(t->left);
		cout<<t->data<<", ";
		traverse_inorder(t->right);
	}

}
```