Question:
Design and implement binary tree and demonstrate its working

Aim:
To Implement Binary Trees

Algorithm:

In-order Traversal:

- For each call check if the corresponding node is NULL , return if It is.
- Call the function recursively for it's left child
- Print the value at stored at corresponding node
- Call the function recursively for it's right child

Pre-Order Traversal:

- For each call check if the corresponding node is NULL , return if It is.
- Print the value at stored at corresponding node
- Call the function recursively for it's left child
- Call the function recursively for it's right child

Pre-Order Traversal:

- For each call check if the corresponding node is NULL , return if It is.
- Call the function recursively for it's left child
- Call the function recursively for it's right child
- Print the value at stored at corresponding node

createNode:

- Firstly allocate required memory for the node
- Then set it's data to value passed to the function
- Set it's right and left children to NULL
- Return the node created

InsertLeft:

- Set the given Node's left child's data to the value passed
- Return the address of the left child

InsertRight:

- Set the given Node's right child's data to the value passed

- Return the address of the right child

Program

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

struct node
{
    int item;
struct node* left;
struct node* right;
};

// Inorder traversal
void inorderTraversal(struct node* root)
{
if (root == NULL) return;
inorderTraversal(root->left);
printf("%d ->", root->item);
inorderTraversal(root->right);
}

// preorderTraversal traversal
void preorderTraversal(struct node* root)
{
    if (root == NULL) return;
printf("%d ->", root->item);
preorderTraversal(root->left);
preorderTraversal(root->right);
}

// postorderTraversal traversal
void postorderTraversal(struct node* root)
{
if (root == NULL) return;
postorderTraversal(root->left);
postorderTraversal(root->right);
printf("%d ->", root->item);
}
```

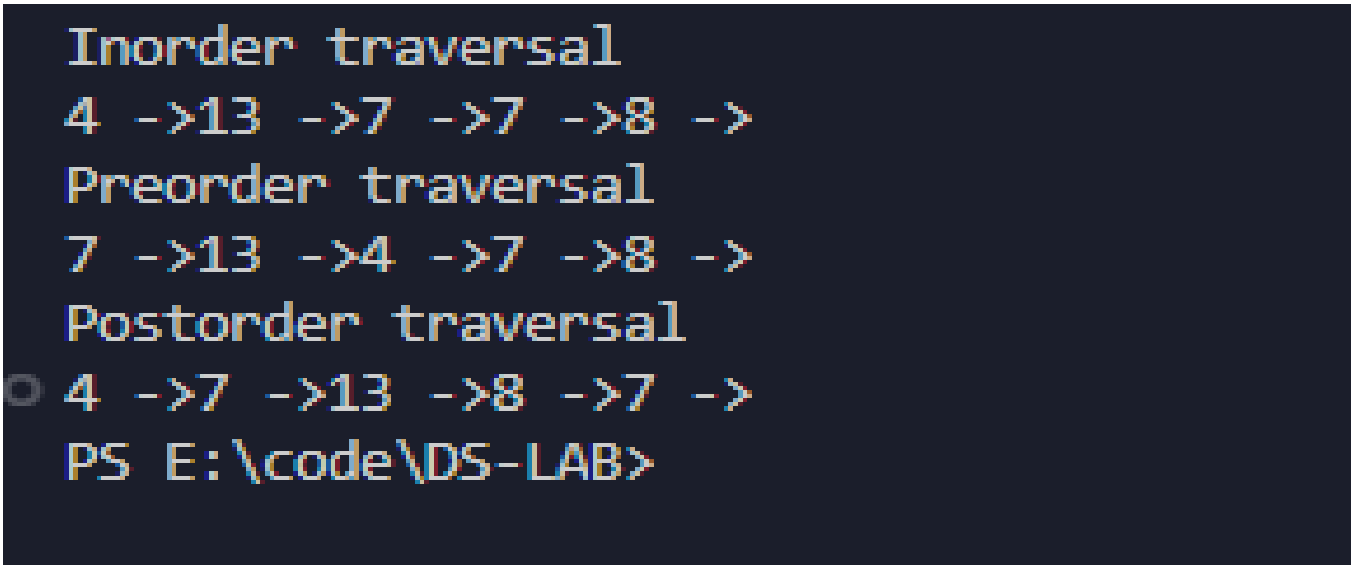
```c
// Creates a new Node
struct node* createNode(int value)
{
struct node* newNode = malloc(sizeof(struct node));
newNode->item = value;
newNode->left = NULL;
newNode->right = NULL;
return newNode;
}

// Insert the left child for given node
struct node* insertLeft(struct node* root, int value)
{
root->left = createNode(value);
return root->left;
}

// Insert the right child for given node
struct node* insertRight(struct node* root, int value)
{
root->right = createNode(value);
return root->right;
}

int main()
{
struct node* root = createNode(7);
insertLeft(root, 13);
insertRight(root, 8);
insertLeft(root->left, 4);
insertRight(root->left, 7);
printf("Inorder traversal \n");
inorderTraversal(root);
printf("\nPreorder traversal \n");
preorderTraversal(root);
printf("\nPostorder traversal \n");
postorderTraversal(root);
}
```

Output:


Inorder traversal
4 ->13 ->7 ->7 ->8 ->
Preorder traversal
7 ->13 ->4 ->7 ->8 ->
Postorder traversal
4 ->7 ->13 ->8 ->7 ->
PS E:\code\DS-LAB>