```c
// 14.Write a C program to implement the Tree Traversals (Inorder, Preorder, Postorder)
#include <stdio.h>
#include <stdlib.h>
struct node {
   int value;
   struct node* left;
   struct node* right;
};
// Inorder traversal
void InOrder(struct node* root) {
   if (root == NULL) return;
   InOrder(root->left);
   printf("%d ", root->value);
   InOrder(root->right);
}
// PreOrder traversal
void PreOrder(struct node* root) {
  if (root == NULL) return;
  printf("%d ", root->value);
  PreOrder(root->left);
  PreOrder(root->right);
}
// PostOrder traversal
void PostOrder(struct node* root) {
  if (root == NULL) return;
  PostOrder(root->left);
  PostOrder(root->right);
  printf("%d ", root->value);
}
// Create a new Node
struct node* createNode(int value) {
  struct node* newNode = malloc(sizeof(struct node));
  newNode->value = value;
  newNode->left = NULL;
  newNode->right = NULL;
  return newNode;
}
int main() {
  struct node* root = createNode(1);
  root->left = createNode(2);
  root->right = createNode(3);
  root->left->left = createNode(4);
  root->left->right = createNode(5);
```

root->right->left = createNode(6);
root->right->right = createNode(7);
printf("Inorder traversal:\t");
InOrder(root);
printf("\PreOrder traversal:\t");
PreOrder(root);
printf("\nPostOrder traversal:\t");
PostOrder(root);
}

```c
// 14.Write a C program to implement the Tree Traversals (Inorder, Preorder, Postorder)
#include <stdio.h>
#include <stdlib.h>
struct node {
    int value;
    struct node* left;
    struct node* right;
};
// Inorder traversal
void InOrder(struct node* root) {
    if (root == NULL) return;
    InOrder(root->left);
    printf("%d ", root->value);
    InOrder(root->right);
}
// PreOrder traversal
void PreOrder(struct node* root) {
    if (root == NULL) return;
    printf("%d ", root->value);
    PreOrder(root->left);
    PreOrder(root->right);
}
// PostOrder traversal
void PostOrder(struct node* root) {
    if (root == NULL) return;
    PostOrder(root->left);
    PostOrder(root->right);
    printf("%d ", root->value);
}
// Create a new Node
struct node* createNode(int value) {
    struct node* newNode = malloc(sizeof(struct node));
    newNode->value = value;
    newNode->left = NULL;
    newNode->right = NULL;
```

```c
        printf("%d ", root->value);
        PreOrder(root->left);
        PreOrder(root->right);
}
    // PostOrder traversal
void PostOrder(struct node* root) {
        if (root == NULL) return;
        PostOrder(root->left);
        PostOrder(root->right);
        printf("%d ", root->value);
}
    // Create a new Node
struct node* createNode(int value) {
        struct node* newNode = malloc(sizeof(struct node));
        newNode->value = value;
        newNode->left = NULL;
        newNode->right = NULL;
        return newNode;
}
int main() {
        struct node* root = createNode(1);
        root->left = createNode(2);
        root->right = createNode(3);
        root->left->left = createNode(4);
        root->left->right = createNode(5);
        root->right->left = createNode(6);
        root->right->right = createNode(7);
        printf("Inorder traversal:\t");
        InOrder(root);
        printf("\nPreOrder traversal:\t");
        PreOrder(root);
        printf("\nPostOrder traversal:\t");
        PostOrder(root);
}
```

Output console:

```
Inorder traversal:      4 2 5 1 6 3 7  PreOrder traversal:      1 2 4 5 3 6 7
PostOrder traversal:    4 5 2 6 7 3 1
--------------------------------
Process exited after 5.734 seconds with return value 2
Press any key to continue . . .
```