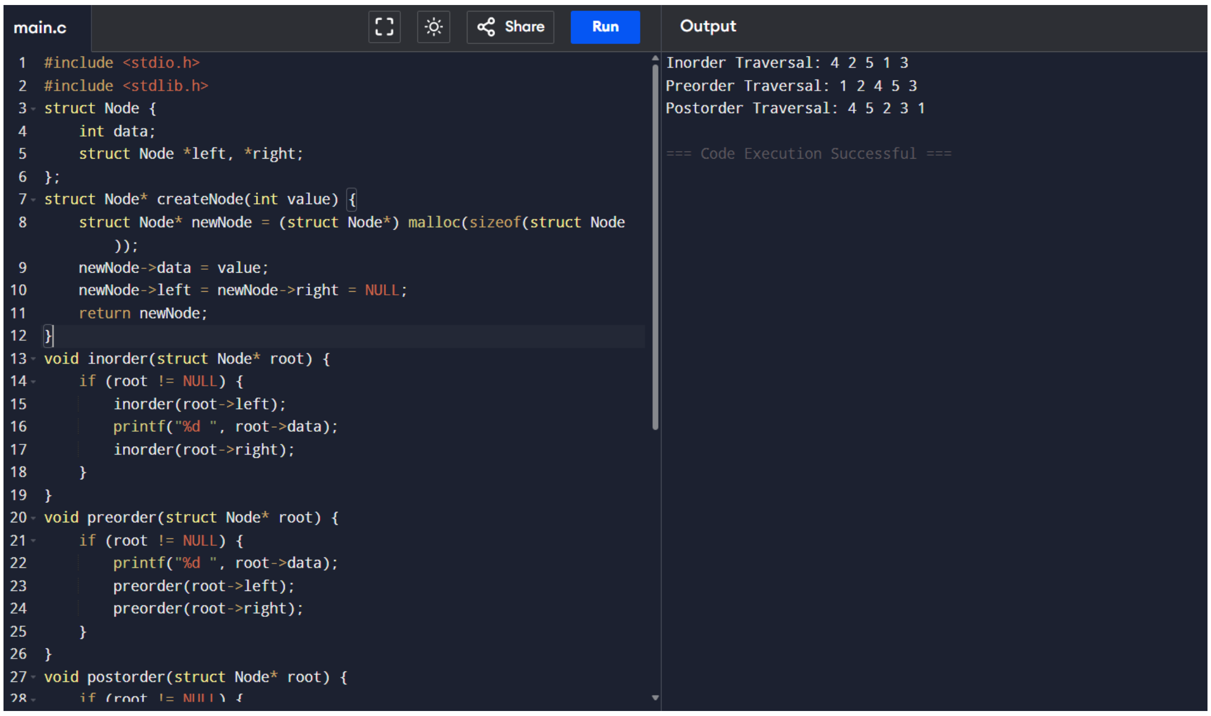
14.Write a C program to implement the Tree Traversals (Inorder, Preorder, Postorder)

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node *left, *right;
};
struct Node* createNode(int value) {
   struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->left = newNode->right = NULL;
   return newNode;
}
void inorder(struct Node* root) {
   if (root != NULL) {
      inorder(root->left);
      printf("%d ", root->data);
      inorder(root->right);
   }
}
void preorder(struct Node* root) {
   if (root != NULL) {
      printf("%d ", root->data);
      preorder(root->left);
      preorder(root->right);
   }
}
void postorder(struct Node* root) {
   if (root != NULL) {
      postorder(root->left);
```

```
        postorder(root->right);
        printf("%d ", root->data);
    }
}
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);
    printf("Inorder Traversal: ");
    inorder(root);
    printf("\nPreorder Traversal: ");
    preorder(root);
    printf("\nPostorder Traversal: ");
    postorder(root);
    return 0;
}
```



```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *left, *right;
};
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node
        ));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}
void preorder(struct Node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
void postorder(struct Node* root) {
    if (root != NULL) {
```

```
Inorder Traversal: 4 2 5 1 3
Preorder Traversal: 1 2 4 5 3
Postorder Traversal: 4 5 2 3 1

=== Code Execution Successful ===
```