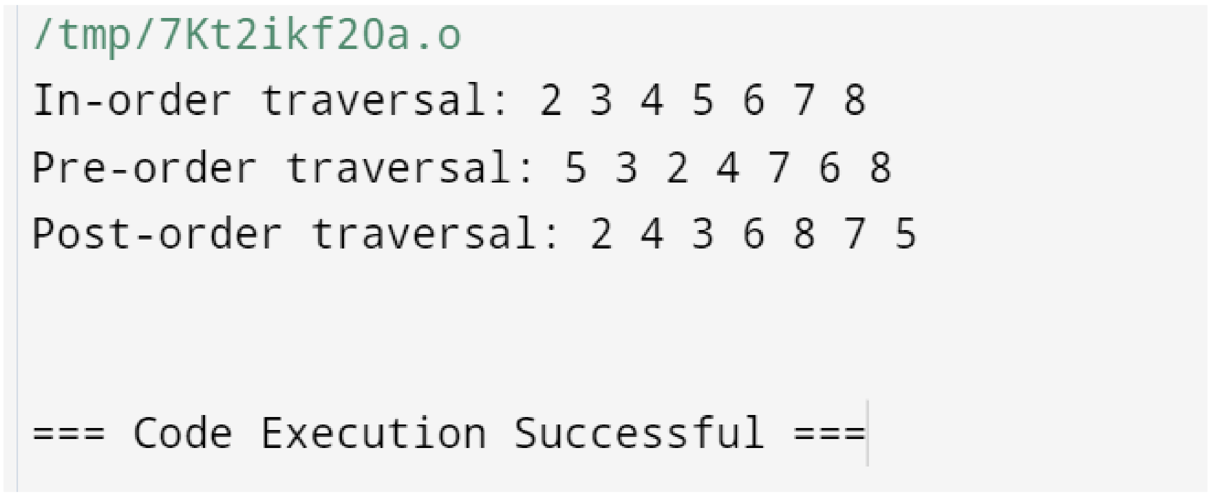
1) **C program to implement a binary tree and perform in-order, pre-order, and post-order traversal**

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Node
{
    int data;
    struct Node* left;
    struct Node* right;
} Node;
Node* createNode(int data)
{
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL)
    {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
Node* insertNode(Node* root, int data)
{
    if (root == NULL)
    {
        return createNode(data);
    }
    if (data < root->data)
    {
        root->left = insertNode(root->left, data);
    }
    else
    {
        root->right = insertNode(root->right, data);
    }
    return root;
}
void inOrderTraversal(Node* root)
{
    if (root != NULL)
    {
        inOrderTraversal(root->left);
        printf("%d ", root->data);
        inOrderTraversal(root->right);
    }
}
void preOrderTraversal(Node* root)
{
    if (root != NULL)
    {
        printf("%d ", root->data);
        preOrderTraversal(root->left);
        preOrderTraversal(root->right);
    }
}
void postOrderTraversal(Node* root)
```

```
{
    if (root != NULL)
    {
        postOrderTraversal(root->left);
        postOrderTraversal(root->right);
        printf("%d ", root->data);
    }
}
void freeTree(Node* root)
{
    if (root != NULL)
    {
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}
int main()
{
    Node* root = NULL;
    root = insertNode(root, 5);
    root = insertNode(root, 3);
    root = insertNode(root, 7);
    root = insertNode(root, 2);
    root = insertNode(root, 4);
    root = insertNode(root, 6);
    root = insertNode(root, 8);
    printf("In-order traversal: ");
    inOrderTraversal(root);
    printf("\n");
    printf("Pre-order traversal: ");
    preOrderTraversal(root);
    printf("\n");
    printf("Post-order traversal: ");
    postOrderTraversal(root);
    printf("\n");
    freeTree(root);
    return 0;
}
```

OUT PUT:



```
/tmp/7Kt2ikf2Oa.o
In-order traversal: 2 3 4 5 6 7 8
Pre-order traversal: 5 3 2 4 7 6 8
Post-order traversal: 2 4 3 6 8 7 5


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