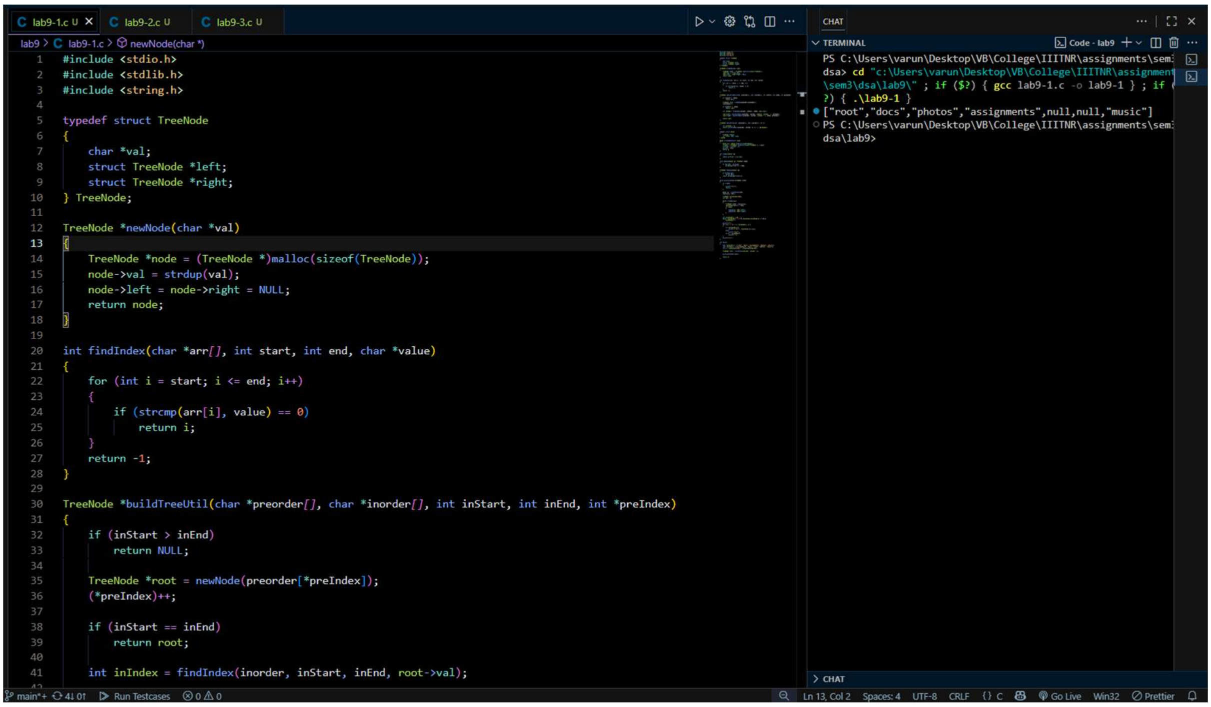
VARUN BHARGAVA – 241010282
DATA STRUCTURES TASK-9

Task 01: Restore File Directory from Creation and Listing Logs:

(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab9/lab9-1.c)
A computer system stores the folder structure of files as a binary tree. Unfortunately, the directory system crashed, and you only have two logs left:

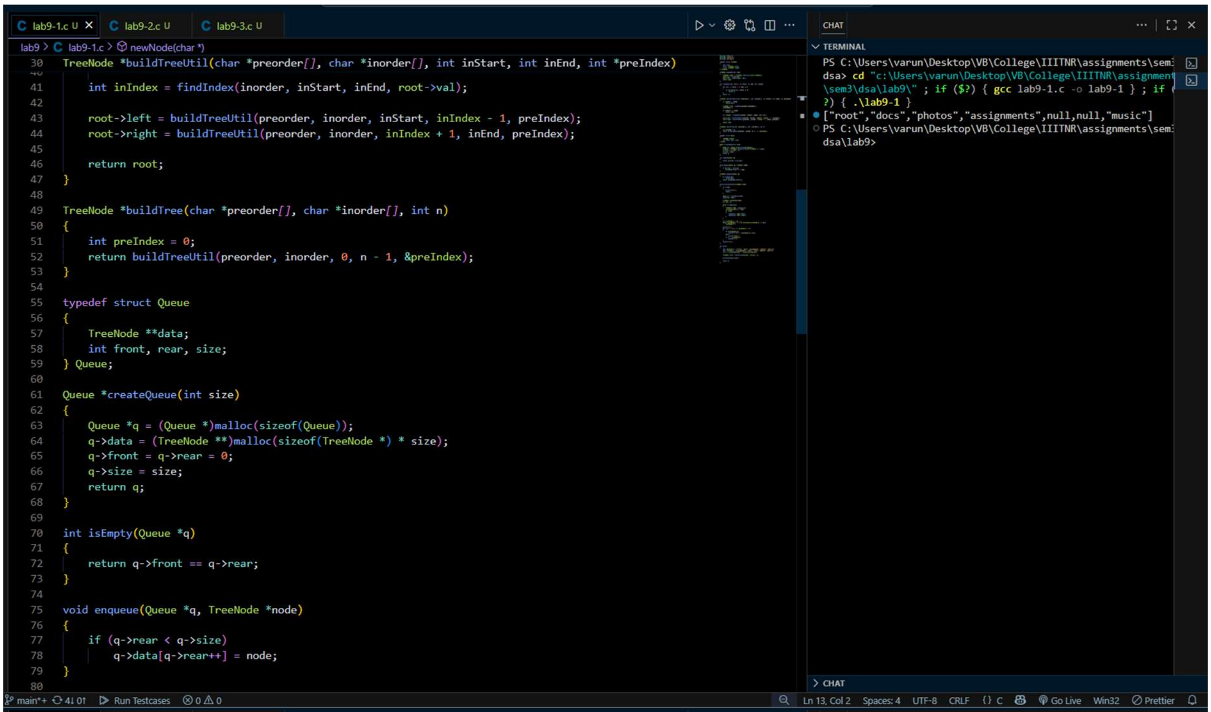
- preorder → the order in which directories and files were created.
- inorder → the order in which files/directories appear when listed alphabetically.

Your task is to rebuild the original folder structure tree and return it.You may assume there are no duplicate directory/file names.
Input:
preorder = ["root","docs","assignments","photos","music"]
inorder = ["assignments","docs","root","photos","music"]
Output:
["root","docs","music","assignments",null,"photos",null]



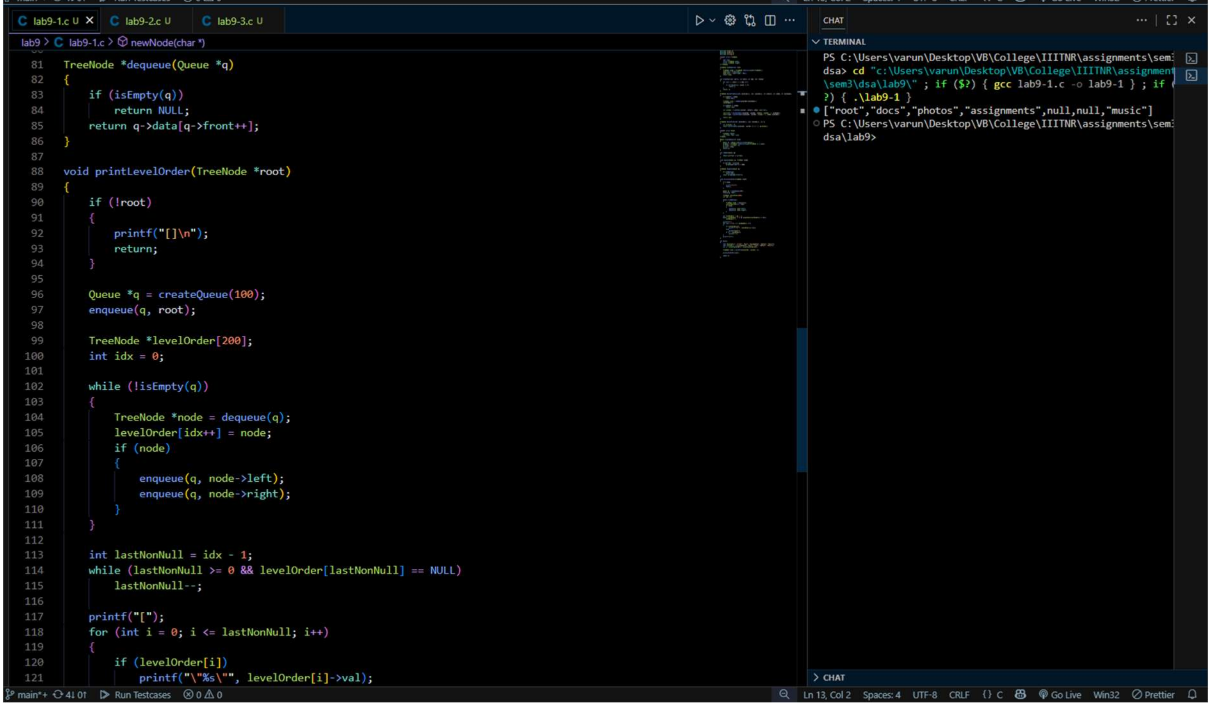
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct TreeNode
{
    char *val;
    struct TreeNode *left;
    struct TreeNode *right;
} TreeNode;

TreeNode *newNode(char *val)
{
    TreeNode *node = (TreeNode *)malloc(sizeof(TreeNode));
    node->val = strdup(val);
    node->left = node->right = NULL;
    return node;
}

int findIndex(char *arr[], int start, int end, char *value)
{
    for (int i = start; i <= end; i++)
    {
        if (strcmp(arr[i], value) == 0)
            return i;
    }
    return -1;
}

TreeNode *buildTreeUtil(char *preorder[], char *inorder[], int inStart, int inEnd, int *preIndex)
{
    if (inStart > inEnd)
        return NULL;

    TreeNode *root = newNode(preorder[*preIndex]);
    (*preIndex)++;

    if (inStart == inEnd)
        return root;

    int inIndex = findIndex(inorder, inStart, inEnd, root->val);
```

```
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignment
\sem3\dsa\lab9\" ; if ($?) { gcc lab9-1.c -o lab9-1 } ; if (
?) { .\lab9-1 }
● ["root","docs","photos","assignments",null,null,"music"]
○ PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```


lab9-1.c U
lab9-2.c U
lab9-3.c U
lab9 > lab9-1.c > newNode(char *)
CHAT
TERMINAL
main*+ 4↓ 0↑ Run Testcases 0 0
Ln 13, Col 2 Spaces: 4 UTF-8 CRLF {} C Go Live Win32 Prettier

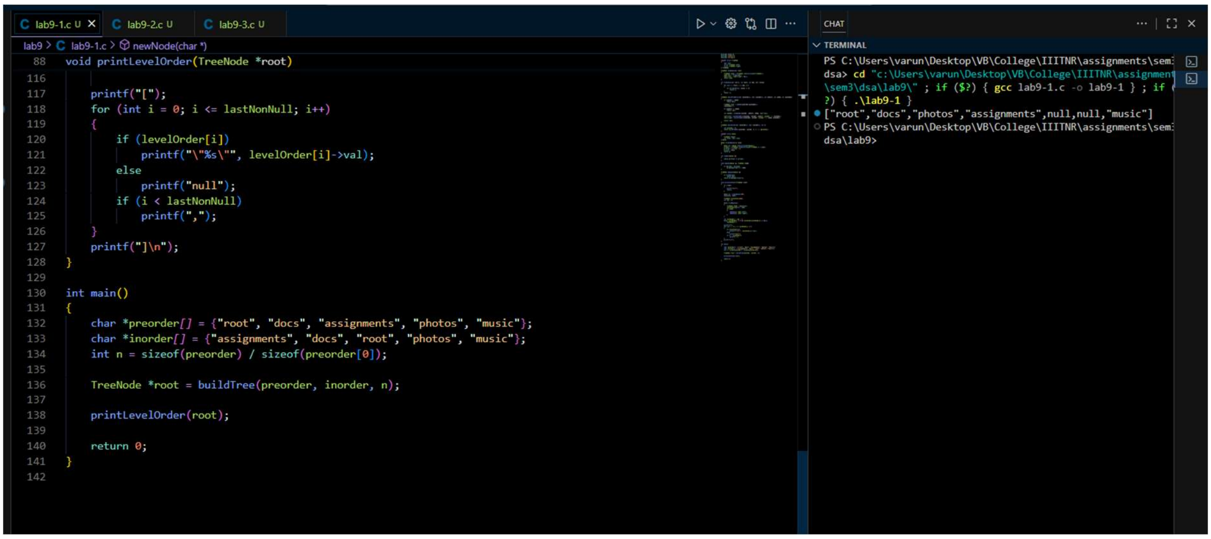
```c
TreeNode *buildTreeUtil(char *preorder[], char *inorder[], int inStart, int inEnd, int *preIndex)
    int inIndex = findIndex(inorder, inStart, inEnd, root->val);

    root->left = buildTreeUtil(preorder, inorder, inStart, inIndex - 1, preIndex);
    root->right = buildTreeUtil(preorder, inorder, inIndex + 1, inEnd, preIndex);

    return root;
}

TreeNode *buildTree(char *preorder[], char *inorder[], int n)
{
    int preIndex = 0;
    return buildTreeUtil(preorder, inorder, 0, n - 1, &preIndex);
}

typedef struct Queue
{
    TreeNode **data;
    int front, rear, size;
} Queue;

Queue *createQueue(int size)
{
    Queue *q = (Queue *)malloc(sizeof(Queue));
    q->data = (TreeNode **)malloc(sizeof(TreeNode *) * size);
    q->front = q->rear = 0;
    q->size = size;
    return q;
}

int isEmpty(Queue *q)
{
    return q->front == q->rear;
}

void enqueue(Queue *q, TreeNode *node)
{
    if (q->rear < q->size)
        q->data[q->rear++] = node;
}
```

```
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignment
\sem3\dsa\lab9\" ; if ($?) { gcc lab9-1.c -o lab9-1 } ; if (
?) { .\lab9-1 }
["root","docs","photos","assignments",null,null,"music"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```


lab9-1.c U
lab9-2.c U
lab9-3.c U
lab9 > lab9-1.c > newNode(char *)
CHAT
TERMINAL
main*+ 4↓ 0↑ Run Testcases 0 0
Ln 13, Col 2 Spaces: 4 UTF-8 CRLF {} C Go Live Win32 Prettier

```c
TreeNode *dequeue(Queue *q)
{
    if (isEmpty(q))
        return NULL;
    return q->data[q->front++];
}

void printLevelOrder(TreeNode *root)
{
    if (!root)
    {
        printf("[]\n");
        return;
    }

    Queue *q = createQueue(100);
    enqueue(q, root);

    TreeNode *levelOrder[200];
    int idx = 0;

    while (!isEmpty(q))
    {
        TreeNode *node = dequeue(q);
        levelOrder[idx++] = node;
        if (node)
        {
            enqueue(q, node->left);
            enqueue(q, node->right);
        }
    }

    int lastNonNull = idx - 1;
    while (lastNonNull >= 0 && levelOrder[lastNonNull] == NULL)
        lastNonNull--;

    printf("[");
    for (int i = 0; i <= lastNonNull; i++)
    {
        if (levelOrder[i])
            printf("\"%s\"", levelOrder[i]->val);
```

```
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignment
\sem3\dsa\lab9\" ; if ($?) { gcc lab9-1.c -o lab9-1 } ; if (
?) { .\lab9-1 }
["root","docs","photos","assignments",null,null,"music"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```



```c
void printLevelOrder(TreeNode *root)

    printf("[");
    for (int i = 0; i <= lastNonNull; i++)
    {
        if (levelOrder[i])
            printf("\"%s\"", levelOrder[i]->val);
        else
            printf("null");
        if (i < lastNonNull)
            printf(",");
    }
    printf("]\n");
}

int main()
{
    char *preorder[] = {"root", "docs", "assignments", "photos", "music"};
    char *inorder[] = {"assignments", "docs", "root", "photos", "music"};
    int n = sizeof(preorder) / sizeof(preorder[0]);

    TreeNode *root = buildTree(preorder, inorder, n);

    printLevelOrder(root);

    return 0;
}
```

```
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assignment
\sem3\dsa\lab9\" ; if ($?) { gcc lab9-1.c -o lab9-1 } ; if (
?) { .\lab9-1 }
["root","docs","photos","assignments",null,null,"music"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```

Task 02: Restore Family Tree from Birth Records:

(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab9/lab9-2.c)

A genealogy company stores family data in two forms:

- inorder → listing of people in chronological order of birth within families.
- postorder → listing of children before their parents.

The database was corrupted, and you must reconstruct the original family tree.

You may assume all names are unique.

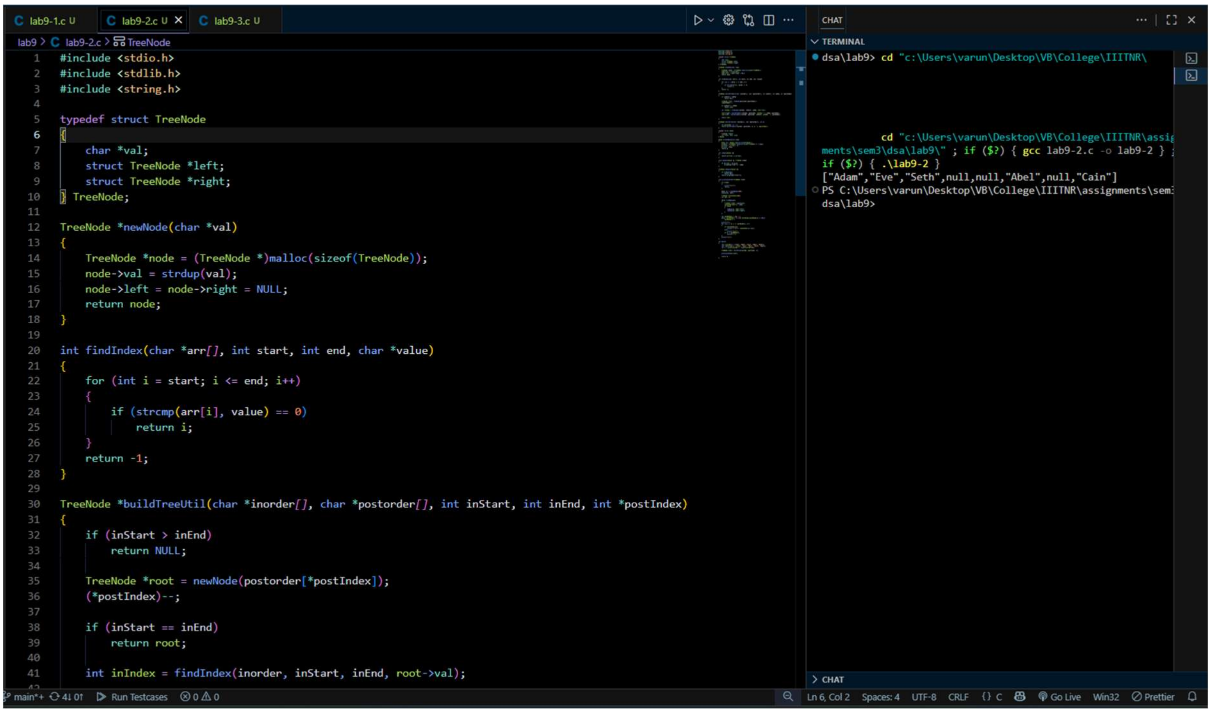
Input:

inorder = ["Eve","Adam","Cain","Abel","Seth"]

postorder = ["Eve","Cain","Abel","Seth","Adam"]

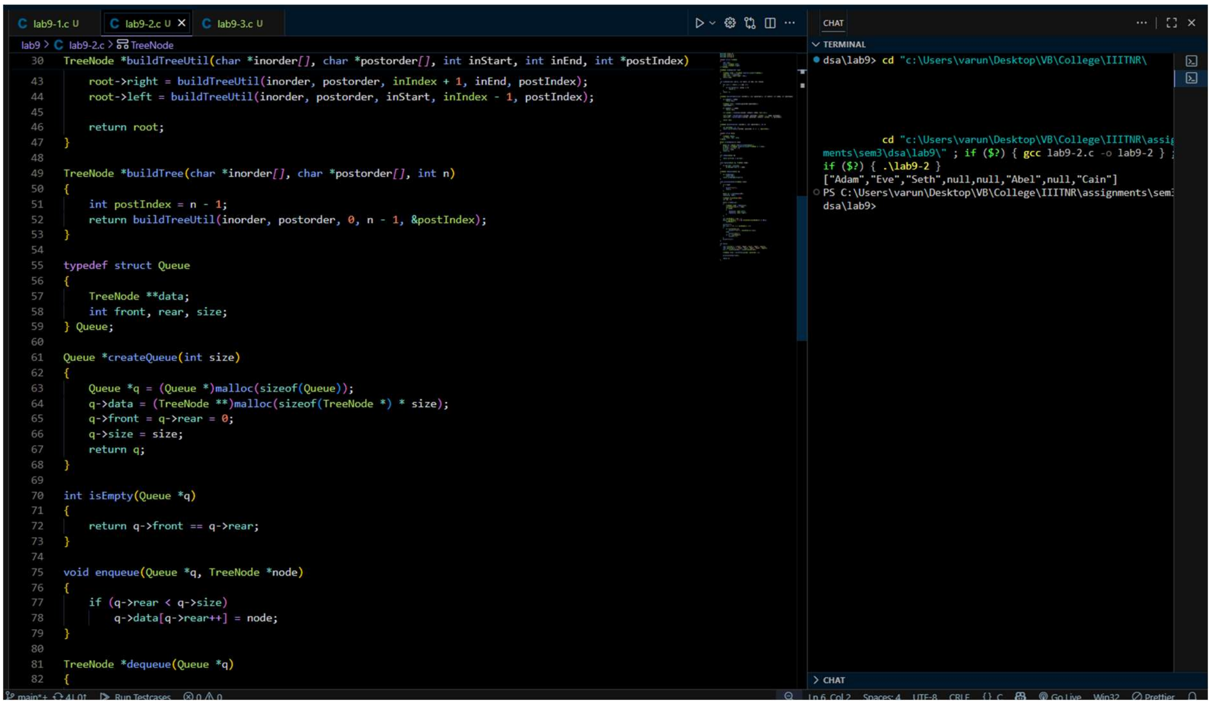
Output:

["Adam","Eve","Seth",null,null,"Cain","Abel"]



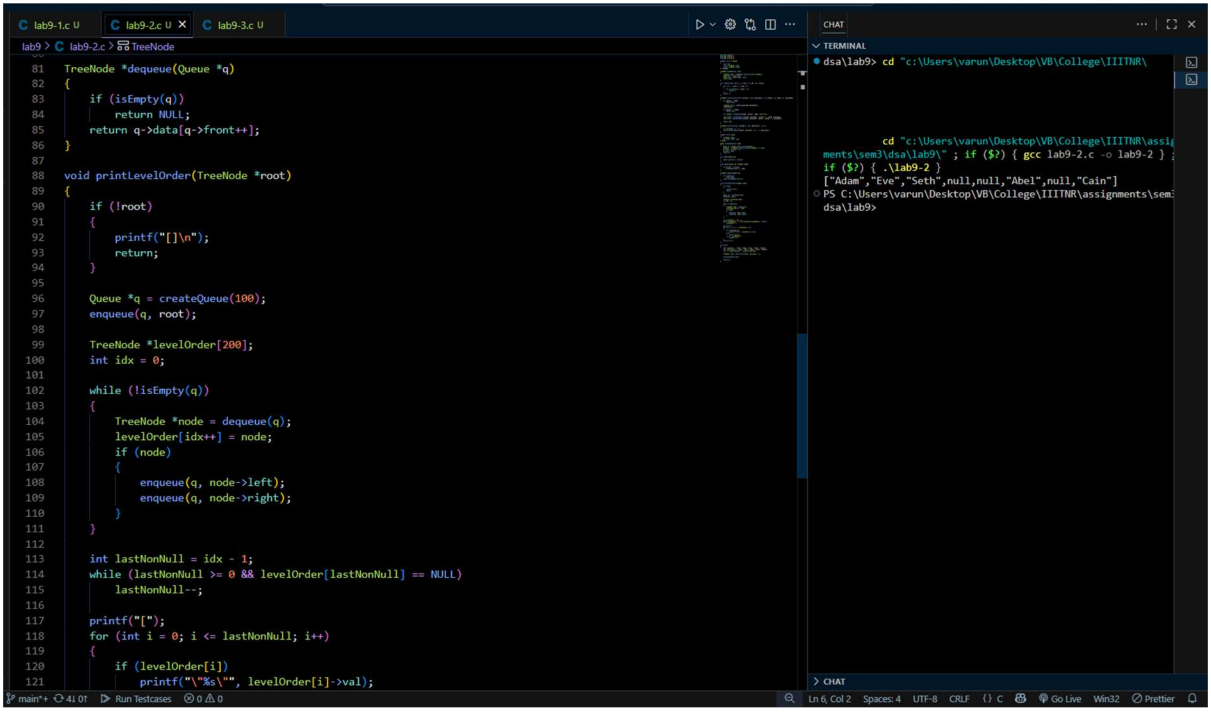
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct TreeNode
{
    char *val;
    struct TreeNode *left;
    struct TreeNode *right;
} TreeNode;

TreeNode *newNode(char *val)
{
    TreeNode *node = (TreeNode *)malloc(sizeof(TreeNode));
    node->val = strdup(val);
    node->left = node->right = NULL;
    return node;
}

int findIndex(char *arr[], int start, int end, char *value)
{
    for (int i = start; i <= end; i++)
    {
        if (strcmp(arr[i], value) == 0)
            return i;
    }
    return -1;
}

TreeNode *buildTreeUtil(char *inorder[], char *postorder[], int inStart, int inEnd, int *postIndex)
{
    if (inStart > inEnd)
        return NULL;

    TreeNode *root = newNode(postorder[*postIndex]);
    (*postIndex)--;

    if (inStart == inEnd)
        return root;

    int inIndex = findIndex(inorder, inStart, inEnd, root->val);
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\




          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-2.c -o lab9-2 } ;
if ($?) { .\lab9-2 }
["Adam","Eve","Seth",null,null,"Abel",null,"Cain"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```



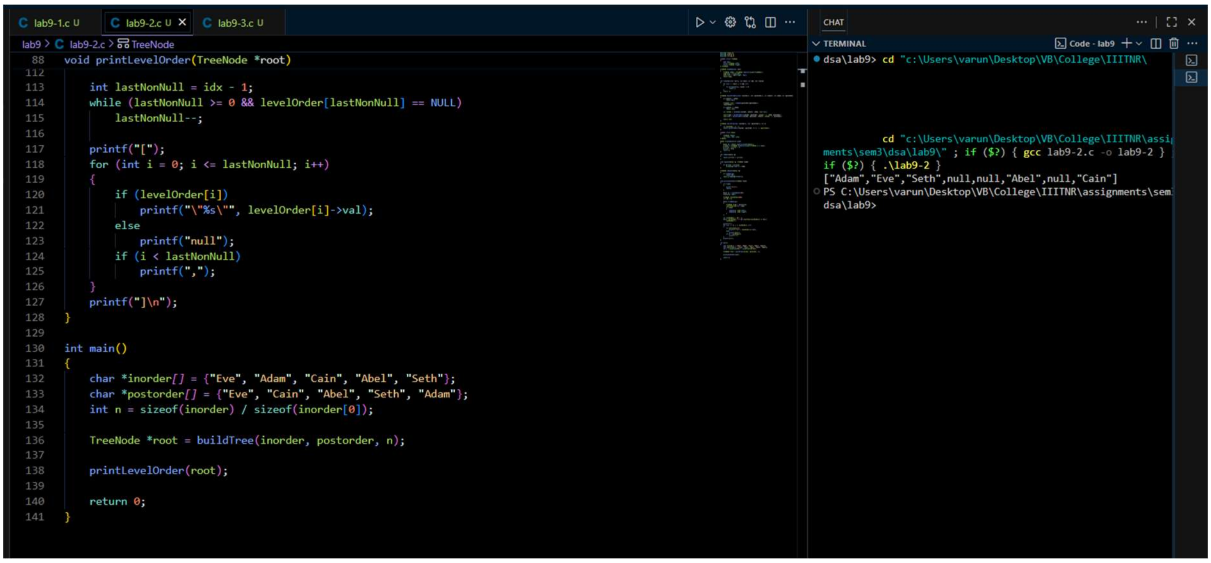
```c
TreeNode *buildTreeUtil(char *inorder[], char *postorder[], int inStart, int inEnd, int *postIndex)
    root->right = buildTreeUtil(inorder, postorder, inIndex + 1, inEnd, postIndex);
    root->left = buildTreeUtil(inorder, postorder, inStart, inIndex - 1, postIndex);

    return root;
}

TreeNode *buildTree(char *inorder[], char *postorder[], int n)
{
    int postIndex = n - 1;
    return buildTreeUtil(inorder, postorder, 0, n - 1, &postIndex);
}

typedef struct Queue
{
    TreeNode **data;
    int front, rear, size;
} Queue;

Queue *createQueue(int size)
{
    Queue *q = (Queue *)malloc(sizeof(Queue));
    q->data = (TreeNode **)malloc(sizeof(TreeNode *) * size);
    q->front = q->rear = 0;
    q->size = size;
    return q;
}

int isEmpty(Queue *q)
{
    return q->front == q->rear;
}

void enqueue(Queue *q, TreeNode *node)
{
    if (q->rear < q->size)
        q->data[q->rear++] = node;
}

TreeNode *dequeue(Queue *q)
{
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\



          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-2.c -o lab9-2 } ;
if ($?) { .\lab9-2 }
["Adam","Eve","Seth",null,null,"Abel",null,"Cain"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```



```c
TreeNode *dequeue(Queue *q)
{
    if (isEmpty(q))
        return NULL;
    return q->data[q->front++];
}

void printLevelOrder(TreeNode *root)
{
    if (!root)
    {
        printf("[]\n");
        return;
    }

    Queue *q = createQueue(100);
    enqueue(q, root);

    TreeNode *levelOrder[200];
    int idx = 0;

    while (!isEmpty(q))
    {
        TreeNode *node = dequeue(q);
        levelOrder[idx++] = node;
        if (node)
        {
            enqueue(q, node->left);
            enqueue(q, node->right);
        }
    }

    int lastNonNull = idx - 1;
    while (lastNonNull >= 0 && levelOrder[lastNonNull] == NULL)
        lastNonNull--;

    printf("[");
    for (int i = 0; i <= lastNonNull; i++)
    {
        if (levelOrder[i])
            printf("\"%s\"", levelOrder[i]->val);
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\



          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-2.c -o lab9-2 } ;
if ($?) { .\lab9-2 }
["Adam","Eve","Seth",null,null,"Abel",null,"Cain"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```



```c
void printLevelOrder(TreeNode *root)

    int lastNonNull = idx - 1;
    while (lastNonNull >= 0 && levelOrder[lastNonNull] == NULL)
        lastNonNull--;

    printf("[");
    for (int i = 0; i <= lastNonNull; i++)
    {
        if (levelOrder[i])
            printf("\"%s\"", levelOrder[i]->val);
        else
            printf("null");
        if (i < lastNonNull)
            printf(",");
    }
    printf("]\n");
}

int main()
{
    char *inorder[] = {"Eve", "Adam", "Cain", "Abel", "Seth"};
    char *postorder[] = {"Eve", "Cain", "Abel", "Seth", "Adam"};
    int n = sizeof(inorder) / sizeof(inorder[0]);

    TreeNode *root = buildTree(inorder, postorder, n);

    printLevelOrder(root);

    return 0;
}
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\



          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assi
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-2.c -o lab9-2 }
if ($?) { .\lab9-2 }
["Adam","Eve","Seth",null,null,"Abel",null,"Cain"]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem
dsa\lab9>
```

Task 03: Reconstruct Network Topology from Connection Logs:

(https://github.com/varunnnb/dsa-sem3-iiitnr/blob/main/lab9/lab9-3.c)

In a distributed network, servers and routers form a tree topology.

- preorder → the order in which connections were established from the main server outward.
- postorder → the order in which messages were acknowledged back to the main server.

Given these two logs, reconstruct the network topology.

If the network is not a full binary tree (where every router has either 0 or 2 connections), the topology cannot be uniquely reconstructed.

Input:

preorder = [1,2,4,5,3,6,7]

postorder = [4,5,2,6,7,3,1]

Output:

[1,2,3,4,5,6,7]

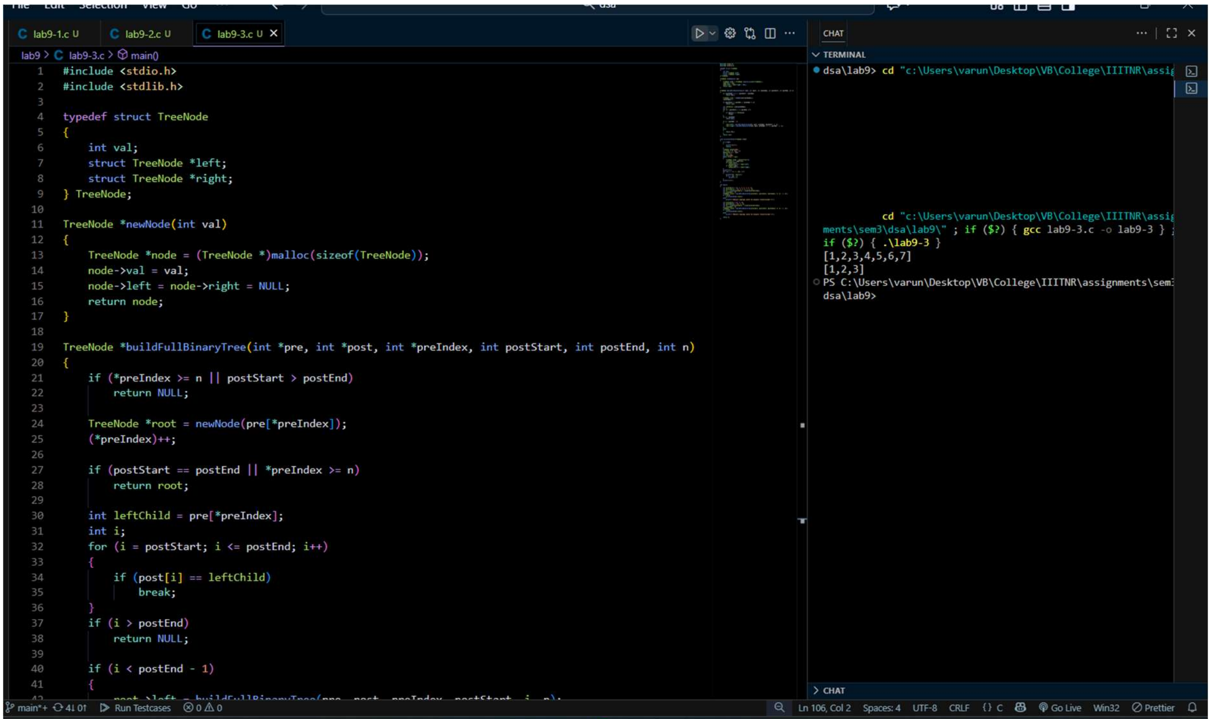
Input:

preorder = [1,2,3]

postorder = [2,3,1]

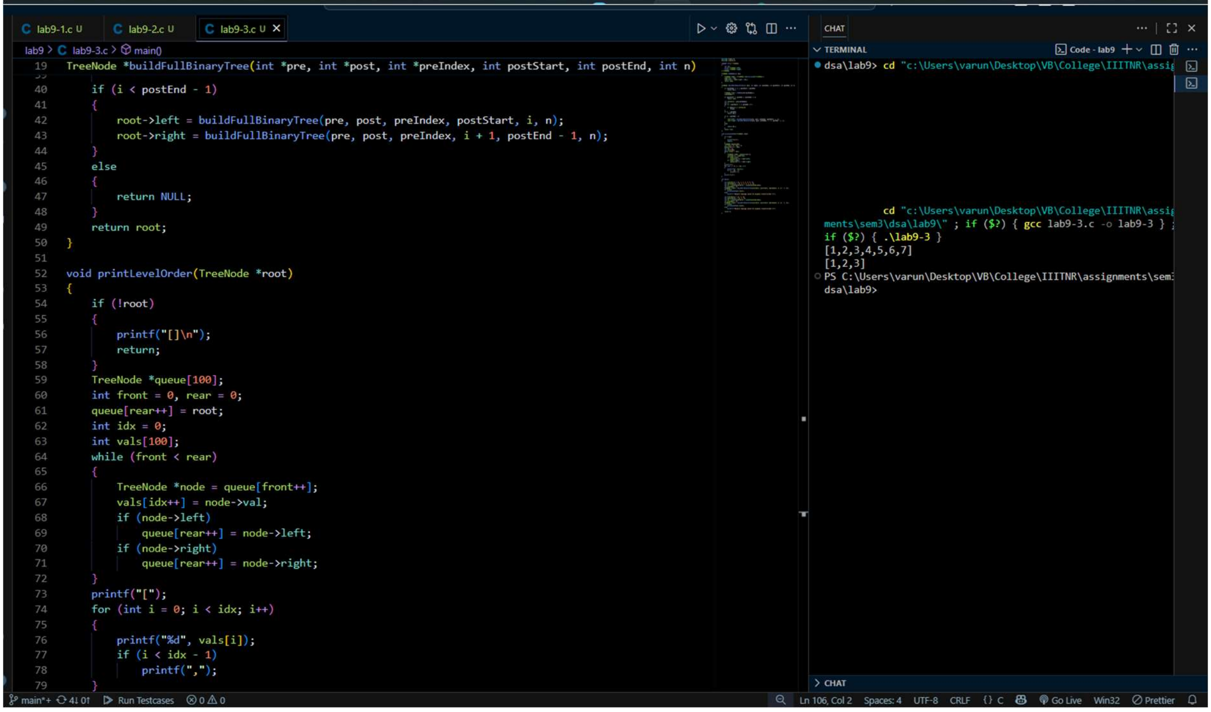
Output:

"Network topology cannot be uniquely reconstructed"



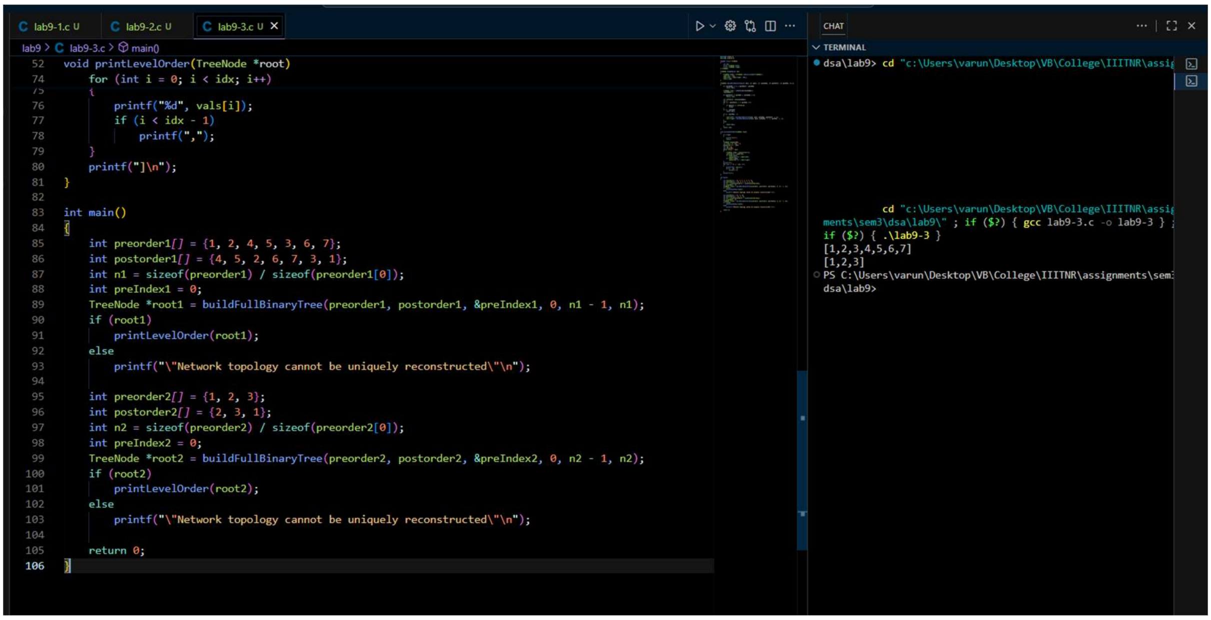
```c
#include <stdio.h>
#include <stdlib.h>

typedef struct TreeNode
{
    int val;
    struct TreeNode *left;
    struct TreeNode *right;
} TreeNode;

TreeNode *newNode(int val)
{
    TreeNode *node = (TreeNode *)malloc(sizeof(TreeNode));
    node->val = val;
    node->left = node->right = NULL;
    return node;
}

TreeNode *buildFullBinaryTree(int *pre, int *post, int *preIndex, int postStart, int postEnd, int n)
{
    if (*preIndex >= n || postStart > postEnd)
        return NULL;

    TreeNode *root = newNode(pre[*preIndex]);
    (*preIndex)++;

    if (postStart == postEnd || *preIndex >= n)
        return root;

    int leftChild = pre[*preIndex];
    int i;
    for (i = postStart; i <= postEnd; i++)
    {
        if (post[i] == leftChild)
            break;
    }
    if (i > postEnd)
        return NULL;

    if (i < postEnd - 1)
    {
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig


          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-3.c -o lab9-3 } ;
if ($?) { .\lab9-3 }
[1,2,3,4,5,6,7]
[1,2,3]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```



```c
TreeNode *buildFullBinaryTree(int *pre, int *post, int *preIndex, int postStart, int postEnd, int n)
    if (i < postEnd - 1)
    {
        root->left = buildFullBinaryTree(pre, post, preIndex, postStart, i, n);
        root->right = buildFullBinaryTree(pre, post, preIndex, i + 1, postEnd - 1, n);
    }
    else
    {
        return NULL;
    }
    return root;
}

void printLevelOrder(TreeNode *root)
{
    if (!root)
    {
        printf("[]\n");
        return;
    }
    TreeNode *queue[100];
    int front = 0, rear = 0;
    queue[rear++] = root;
    int idx = 0;
    int vals[100];
    while (front < rear)
    {
        TreeNode *node = queue[front++];
        vals[idx++] = node->val;
        if (node->left)
            queue[rear++] = node->left;
        if (node->right)
            queue[rear++] = node->right;
    }
    printf("[");
    for (int i = 0; i < idx; i++)
    {
        printf("%d", vals[i]);
        if (i < idx - 1)
            printf(",");
    }
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig


          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-3.c -o lab9-3 } ;
if ($?) { .\lab9-3 }
[1,2,3,4,5,6,7]
[1,2,3]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```



```c
void printLevelOrder(TreeNode *root)
    for (int i = 0; i < idx; i++)
    {
        printf("%d", vals[i]);
        if (i < idx - 1)
            printf(",");
    }
    printf("]\n");
}

int main()
{
    int preorder1[] = {1, 2, 4, 5, 3, 6, 7};
    int postorder1[] = {4, 5, 2, 6, 7, 3, 1};
    int n1 = sizeof(preorder1) / sizeof(preorder1[0]);
    int preIndex1 = 0;
    TreeNode *root1 = buildFullBinaryTree(preorder1, postorder1, &preIndex1, 0, n1 - 1, n1);
    if (root1)
        printLevelOrder(root1);
    else
        printf("\"Network topology cannot be uniquely reconstructed\"\n");

    int preorder2[] = {1, 2, 3};
    int postorder2[] = {2, 3, 1};
    int n2 = sizeof(preorder2) / sizeof(preorder2[0]);
    int preIndex2 = 0;
    TreeNode *root2 = buildFullBinaryTree(preorder2, postorder2, &preIndex2, 0, n2 - 1, n2);
    if (root2)
        printLevelOrder(root2);
    else
        printf("\"Network topology cannot be uniquely reconstructed\"\n");

    return 0;
}
```

```
dsa\lab9> cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig







          cd "c:\Users\varun\Desktop\VB\College\IIITNR\assig
ments\sem3\dsa\lab9\" ; if ($?) { gcc lab9-3.c -o lab9-3 } ;
if ($?) { .\lab9-3 }
[1,2,3,4,5,6,7]
[1,2,3]
PS C:\Users\varun\Desktop\VB\College\IIITNR\assignments\sem3
dsa\lab9>
```