Institute of Computer Technology

B. Tech. Computer Science and Engineering

Sub: DS

Course Code: 2CSE302

Practical – 14

Name: Jaymin Gondaliya

Enrollment No: 23162171007

Sem - 3

Branch: CS

Class: A

Batch: 32

Problem Definition-1: You are given two strings representing the inorder and preorder traversals of a binary tree. Your task is to:

1. Construct the binary tree using these traversals.

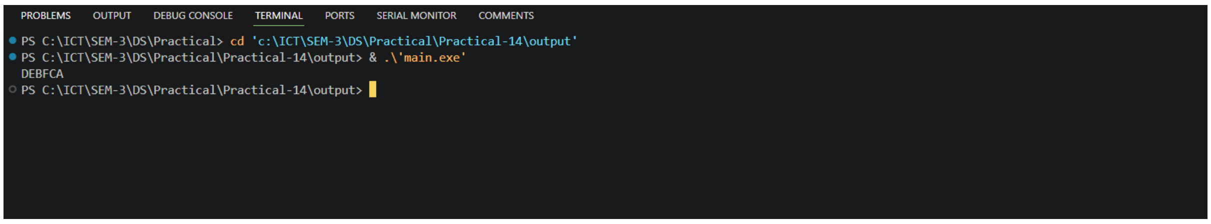
2. Generate the postorder traversal of the constructed binary tree and print it as an output string.

Code:

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

// Define the structure of a node in the binary tree
struct Node {
    char data;                  // The data value (character) of the node
    struct Node* left;          // Pointer to the left child
    struct Node* right;         // Pointer to the right child
};

// Function to create a new node with a given character
struct Node* newNode(char data) {
    // Allocate memory for a new node
    struct Node* node = (struct Node*)malloc(sizeof(struct Node));
    node->data = data;          // Set the node's data
    node->left = NULL;          // Initialize left child as NULL
    node->right = NULL;         // Initialize right child as NULL
    return node;                // Return the created node
}

// Function to find the index of a given character in the inorder array
int findIndex(char* inorder, int start, int end, char value) {
    for (int i = start; i <= end; i++) {    // Loop through the inorder array
        if (inorder[i] == value) {          // If the character matches the
value
            return i;                       // Return its index
        }
    }
    return -1;  // This won't happen as the character is guaranteed to exist
}

// Function to build the binary tree from inorder and preorder arrays
struct Node* buildTree(char* inorder, char* preorder, int* preIndex, int
start, int end) {
    if (start > end) return NULL;           // If no elements are left, return
NULL

    // Get the next value from preorder as the root of the current subtree
    char current = preorder[*preIndex];
    (*preIndex)++;                          // Move to the next element in
preorder

    // Create a new node for the root
    struct Node* node = newNode(current);

    // If there's only one node (leaf node), return it
    if (start == end) return node;

    // Find the index of the current node's value in the inorder array
    int inIndex = findIndex(inorder, start, end, current);

    // Recursively build the left subtree (from start to inIndex-1)
    node->left = buildTree(inorder, preorder, preIndex, start, inIndex - 1);

    // Recursively build the right subtree (from inIndex+1 to end)
    node->right = buildTree(inorder, preorder, preIndex, inIndex + 1, end);

    return node;  // Return the built node
}

// Function to print the postorder traversal of the binary tree
```

```c
void postorderTraversal(struct Node* root) {
    if (root == NULL) return;                // If the node is NULL, return

    postorderTraversal(root->left);          // Traverse the left subtree
    postorderTraversal(root->right);         // Traverse the right subtree
    printf("%c", root->data);                // Print the root (postorder)
}

// Main function to build the tree and print its postorder traversal
int main() {
    char inorder[] = "DBEACF";               // Inorder traversal input
    char preorder[] = "ABDECF";              // Preorder traversal input
    int n = 6;                               // Length of the inorder string
    int preIndex = 0;                        // Initial index for preorder array

    // Build the binary tree from the traversals
    struct Node* root = buildTree(inorder, preorder, &preIndex, 0, n - 1);

    // Print the postorder traversal of the constructed tree
    postorderTraversal(root);                // Expected output: DEBFCA
    printf("\n");                            // Print a newline at the end



    return 0;
}
```

Output:



```
PS C:\ICT\SEM-3\DS\Practical> cd 'c:\ICT\SEM-3\DS\Practical\Practical-14\output'
PS C:\ICT\SEM-3\DS\Practical\Practical-14\output> & .\'main.exe'
DEBFCA
PS C:\ICT\SEM-3\DS\Practical\Practical-14\output>
```