**Institute of Computer Technology**

**B. Tech. Computer Science and Engineering**

**Sub: DS**

**Course Code: 2CSE302**

**Practical – 15**

**Name: Jaymin Gondaliya**

**Enrollment No: 23162171007**

**Sem - 3**

**Branch: CS**

**Class: A**

**Batch: 32**

**Problem Definition-1:** Imagine you have a larger binary tree representing a company organizational structure, and within it, you want to check if there a smaller binary tree that represents a specific department. Your tasked with determining if the department tree is indeed a part of the larger organizational tree. To solve this, you use a C program that performs a recursive search, checking if there is a matching subtree within the larger tree. If found, the program confirms the presence of the department tree within the organizational structure..

**Code:**

*#include* <stdio.h>

*#include* <stdlib.h>

*//* **Define the structure for each tree node**

struct TreeNode {

    int data;*//* **Holds the value of the node**

    struct TreeNode\* left;*//* **Points to the left child node**

    struct TreeNode\* right;*//* **Points to the right child node**

};

*//* **Function to make a new node with given data**

struct TreeNode\* newNode(int *data*) {

    struct TreeNode\* node = (struct TreeNode\*)malloc(sizeof(struct TreeNode));*//* **Create a new node**

    node->data = *data*;*//* **Store the data in the new node**

    node->left = NULL;*//* **Start with no left child**

    node->right = NULL;*//* **Start with no right child**

*return* node;*//* **Return the new node to use in the tree**

}

*//* **Function to insert nodes in a complete tree fashion**

void insert(struct TreeNode\*\* *root*, int *data*) {

    struct TreeNode\* node = newNode(*data*);*//* **Make a new node with given data**

*//* **If the tree is empty, this new node becomes the root**

*if* (\**root* == NULL) {

        \**root* = node;

*return*;

    }

*//* **Create a queue to help find the right spot for the new node**

    struct TreeNode\* queue[100];

    int front = 0, rear = 0;

    queue[rear++] = \**root*;*//* **Start by adding the root to the queue**

*//* **Keep going through the queue until we find a free spot**

*while* (front < rear) {

        struct TreeNode\* temp = queue[front++];*//* **Get the front node in the queue**

*//* **Check if there's room on the left side**

*if* (temp->left == NULL) {

            temp->left = node;*//* **Place the new node as the left child**

*break*;*//* **Done with insertion, so exit the loop**

        } *else* {

            queue[rear++] = temp->left;*//* **Add the left child to queue**

        }

*//* **Check if there's room on the right side**

*if* (temp->right == NULL) {

            temp->right = node;*//* **Place the new node as the right child**

*break*;*//* **Done with insertion, so exit the loop**

        } *else* {

            queue[rear++] = temp->right;*//* **Add the right child to queue**

        }

    }

}

*//* **Function to print out the tree level by level**

void printLevelOrder(struct TreeNode\* *root*) {

*//* **If the tree is empty, say so**

*if* (*root* == NULL) {

        printf("The tree is empty.\n");

*return*;

    }

*//* **Set up a queue to help us print each level**

    struct TreeNode\* queue[100];

    int front = 0, rear = 0;

    queue[rear++] = *root*;*//* **Start with the root in the queue**

    printf("Binary Tree Structure:\n");

*//* **Go through each level of the tree**

*while* (front < rear) {

        int levelSize = rear - front;*//* **Nodes in the current level**

*//* **Print all nodes in this level**

*for* (int i = 0; i < levelSize; i++) {

            struct TreeNode\* temp = queue[front++];*//* **Get node from the front**

            printf("%d ", temp->data);*//* **Print the node's data**

*//* **Add the left child to the queue if it exists**

*if* (temp->left != NULL) queue[rear++] = temp->left;

*//* **Add the right child to the queue if it exists**

*if* (temp->right != NULL) queue[rear++] = temp->right;

        }

        printf("\n");*//* **Go to a new line after each level**

    }

}

int main() {

    struct TreeNode\* root = NULL;*//* **Start with an empty tree**

    int data;

    printf("Binary Tree Creation:\n");

*//* **Get user input to build the tree**

*while* (1) {

        printf("Enter node data (enter -1 to stop): ");

        scanf("%d", &data);*//* **Read the data from the user**

*if* (data == -1) *break*;*//* **Stop if the user enters -1**

        insert(&root, data);*//* **Add the data to the tree**

    }

*//* **Display the tree level by level**

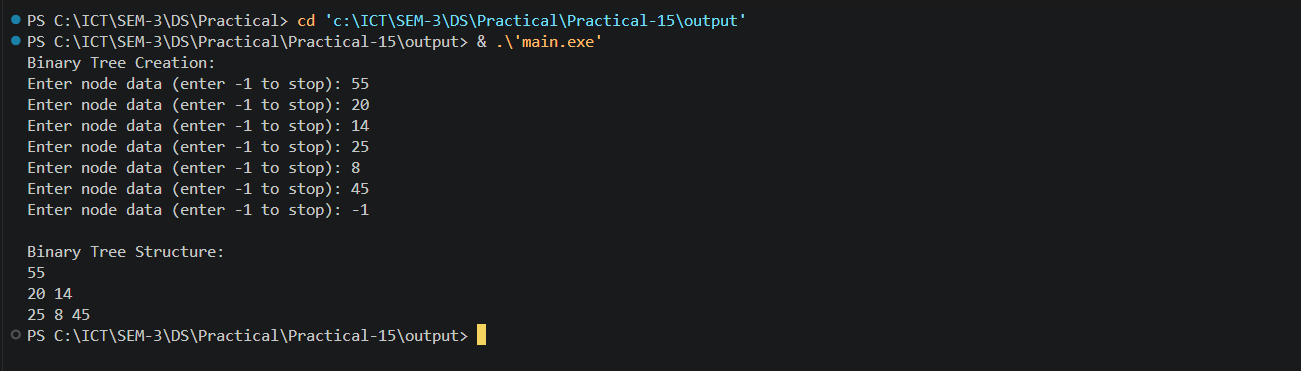
    printf("\n");

    printLevelOrder(root);

*return* 0;

}

**Output:**

****