**Check for Symmetric Binary Tree (Iterative Approach)**

C programming

// C++ program to check if a given Binary

// Tree is symmetric or not

#include<bits/stdc++.h>

using namespace std;

// A Binary Tree Node

struct Node

{

    int key;

    struct Node\* left, \*right;

};

// Utility function to create new Node

Node \*newNode(int key)

{

    Node \*temp = new Node;

    temp->key = key;

    temp->left = temp->right = NULL;

    return (temp);

}

// Returns true if a tree is symmetric

// i.e. mirror image of itself

bool isSymmetric(struct Node\* root)

{

    if(root == NULL)

        return true;

    // If it is a single tree node, then

    // it is a symmetric tree.

    if(!root->left && !root->right)

        return true;

    queue <Node\*> q;

    // Add root to queue two times so that

    // it can be checked if either one child

    // alone is NULL or not.

    q.push(root);

    q.push(root);

    // To store two nodes for checking their

    // symmetry.

    Node\* leftNode, \*rightNode;

    while(!q.empty()){

        // Remove first two nodes to check

        // their symmetry.

        leftNode = q.front();

        q.pop();

        rightNode = q.front();

        q.pop();

        // if both left and right nodes

        // exist, but have different

        // values--> inequality, return false

        if(leftNode->key != rightNode->key){

        return false;

        }

        // Push left child of left subtree node

        // and right child of right subtree

        // node in queue.

        if(leftNode->left && rightNode->right){

            q.push(leftNode->left);

            q.push(rightNode->right);

        }

        // If only one child is present alone

        // and other is NULL, then tree

        // is not symmetric.

        else if (leftNode->left || rightNode->right)

        return false;

        // Push right child of left subtree node

        // and left child of right subtree node

        // in queue.

        if(leftNode->right && rightNode->left){

            q.push(leftNode->right);

            q.push(rightNode->left);

        }

        // If only one child is present alone

        // and other is NULL, then tree

        // is not symmetric.

        else if(leftNode->right || rightNode->left)

        return false;

    }

    return true;

}

// Driver program

int main()

{

    // Let us construct the Tree shown in

    // the above figure

    Node \*root = newNode(1);

    root->left = newNode(2);

    root->right = newNode(2);

    root->left->left = newNode(3);

    root->left->right = newNode(4);

    root->right->left = newNode(4);

    root->right->right = newNode(3);

    if(isSymmetric(root))

        cout << "The given tree is Symmetric";

    else

        cout << "The given tree is not Symmetric";

    return 0;

}