

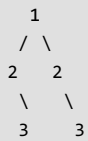
## Symmetric Tree (Mirror Image of itself)

Given a binary tree, check whether it is a mirror of itself.

For example, this binary tree is symmetric:



But the following is not:



**We strongly recommend you to minimize your browser and try this yourself first.**

The idea is to write a recursive function `isMirror()` that takes two trees as argument and returns true if trees are mirror and false if trees are not mirror. The `isMirror()` function recursively checks two roots and subtrees under the root.

Below is implementation of above algorithm.

**C++**

```

// 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 trees with roots as root1 and root2 are mirror
bool isMirror(struct Node *root1, struct Node *root2)
{
    // If both trees are empty, then they are mirror images
    if (root1 == NULL && root2 == NULL)
        return true;

    // For two trees to be mirror images, the following three
    // conditions must be true
    // 1 - Their root node's key must be same
    // 2 - left subtree of left tree and right subtree
    //    of right tree have to be mirror images
    // 3 - right subtree of left tree and left subtree
    //    of right tree have to be mirror images
    if (root1 && root2 && root1->key == root2->key)
        return isMirror(root1->left, root2->right) &&
            isMirror(root1->right, root2->left);

    // if neither of above conditions is true then root1
    // and root2 are not mirror images
    return false;
}

// Returns true if a tree is symmetric i.e. mirror image of itself
bool isSymmetric(struct Node* root)
{
    // Check if tree is mirror of itself
    return isMirror(root, root);
}

// 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);

    cout << isSymmetric(root);
    return 0;
}

```

## Java

```

// Java program to check is binary tree is symmetric or not
class Node
{
    int key;
    Node left, right;

    Node(int item)
    {
        key = item;
        left = right = null;
    }
}

class BinaryTree
{
    Node root;

    // returns true if trees with roots as root1 and root2 are mirror
    boolean isMirror(Node node1, Node node2)
    {
        // if both trees are empty, then they are mirror image
        if (node1 == null && node2 == null)
            return true;

        // For two trees to be mirror images, the following three
        // conditions must be true
        // 1 - Their root node's key must be same
        // 2 - left subtree of left tree and right subtree
        //    of right tree have to be mirror images
        // 3 - right subtree of left tree and left subtree
        //    of right tree have to be mirror images
        if (node1 != null && node2 != null && node1.key == node2.key)
            return (isMirror(node1.left, node2.right)
                    && isMirror(node1.right, node2.left));

        // if neither of the above conditions is true then
        // root1 and root2 are mirror images
        return false;
    }

    // returns true if the tree is symmetric i.e
    // mirror image of itself
    boolean isSymmetric(Node node)
    {
        // check if tree is mirror of itself
        return isMirror(root, root);
    }

    // Driver program
    public static void main(String args[])
    {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(2);
        tree.root.left.left = new Node(3);
        tree.root.left.right = new Node(4);
        tree.root.right.left = new Node(4);
        tree.root.right.right = new Node(3);
        boolean output = tree.isSymmetric(tree.root);
        if (output == true)
            System.out.println("1");
        else
            System.out.println("0");
    }
}

// This code has been contributed by Mayank Jaiswal

```

```

# Python program to check if a given Binary Tree is
# symmetric or not

# Node structure
class Node:

    # Utility function to create new node
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

# Returns True if trees with roots as root1 and root 2
# are mirror
def isMirror(root1 , root2):
    # If both trees are empty, then they are mirror images
    if root1 is None and root2 is None:
        return True

    """ For two trees to be mirror images, the following three
    conditions must be true
    1 - Their root node's key must be same
    2 - left subtree of left tree and right subtree
    of right tree have to be mirror images
    3 - right subtree of left tree and left subtree
    of right tree have to be mirror images
    """
    if (root1 is not None and root2 is not None):
        if root1.key == root2.key:
            return (isMirror(root1.left, root2.right)and
                    isMirror(root1.right, root2.left))

    # If neither of above conditions is true then root1
    # and root2 are not mirror images
    return False

def isSymmetric(root):

    # Check if tree is mirror of itself
    return isMirror(root, root)

# Driver Program
# Let's construct the tree show in the above figure
root = Node(1)
root.left = Node(2)
root.right = Node(2)
root.left.left = Node(3)
root.left.right = Node(4)
root.right.left = Node(4)
root.right.right = Node(3)
print "1" if isSymmetric(root) == True else "0"

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)

```

Output:

1