***BINARY TREE IN js:***

<script>

// Iterative Javascript program to check if

// given binary tree symmetric

var root = null;

class Node

{

constructor(v)

{

this.val = v;

this.left = null;

this.right = null;

}

}

/\* Function to check if the tree is Symmetric \*/

function isSymmetric(root)

{

/\* This allows adding null

elements to the queue \*/

var q = [];

/\* Initially, add left and

right nodes of root \*/

q.push(root.left);

q.push(root.right);

while (q.length != 0)

{

/\* Remove the front 2 nodes to

check for equality \*/

var tempLeft = q[0];

q.shift();

var tempRight = q[0];

q.shift();

/\* If both are null, continue and check

for further elements \*/

if (tempLeft == null && tempRight == null)

continue;

/\* If only one is null---inequality, return false \*/

if ((tempLeft == null && tempRight != null) ||

(tempLeft != null && tempRight == null))

return false;

/\* If both left and right nodes exist, but

have different values-- inequality,

return false\*/

if (tempLeft.val != tempRight.val)

return false;

/\* Note the order of insertion of elements

to the queue :

1) left child of left subtree

2) right child of right subtree

3) right child of left subtree

4) left child of right subtree \*/

q.push(tempLeft.left);

q.push(tempRight.right);

q.push(tempLeft.right);

q.push(tempRight.left);

}

/\* If the flow reaches here, return true\*/

return true;

}

// Driver code

var n = new Node(1);

root = n;

root.left = new Node(2);

root.right = new Node(2);

root.left.left = new Node(3);

root.left.right = new Node(4);

root.right.left = new Node(4);

root.right.right = new Node(3);

if (isSymmetric(root))

document.write("The given tree is Symmetric");

else

document.write("The given tree is not Symmetric");

// This code is contributed by noob2000

</script>