Check if all leaves are at same level

Given a Binary Tree, check if all leaves are at same level or not.

```
        12
      /    \
     5       7
   /          \
  3            1
 Leaves are at same level

        12
      /    \
     5       7
   /
  3
  Leaves are Not at same level


        12
      /
     5
   /   \
  3     9
 /      /
1      2
Leaves are at same level
```

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

The idea is to first find level of the leftmost leaf and store it in a variable leafLevel. Then compare level of all other leaves with leafLevel, if same, return true, else return false. We traverse the given Binary Tree in Preorder fashion. An argument leaflevel is passed to all calls. The value of leafLevel is initialized as 0 to indicate that the first leaf is not yet seen yet. The value is updated when we find first leaf. Level of subsequent leaves (in preorder) is compared with leafLevel.

C

```c
// C program to check if all leaves are at same level
#include <stdio.h>
#include <stdlib.h>

// A binary tree node
struct Node
{
    int data;
    struct Node *left, *right;
};

// A utility function to allocate a new tree node
struct Node* newNode(int data)
{
    struct Node* node = (struct Node*) malloc(sizeof(struct Node));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

/* Recursive function which checks whether all leaves are at same level */
bool checkUtil(struct Node *root, int level, int *leafLevel)
{
    // Base case
    if (root == NULL)  return true;

    // If a leaf node is encountered
    if (root->left == NULL && root->right == NULL)
    {
        // When a leaf node is found first time
        if (*leafLevel == 0)
        {
            *leafLevel = level; // Set first found leaf's level
            return true;
        }

        // If this is not first leaf node, compare its level with
        // first leaf's level
        return (level == *leafLevel);
    }

    // If this node is not leaf, recursively check left and right subtrees
    return checkUtil(root->left, level+1, leafLevel) &&
           checkUtil(root->right, level+1, leafLevel);
}

/* The main function to check if all leafs are at same level.
   It mainly uses checkUtil() */
bool check(struct Node *root)
{
   int level = 0, leafLevel = 0;
   return checkUtil(root, level, &leafLevel);
}

// Driver program to test above function
int main()
{
    // Let us create tree shown in thirdt example
    struct Node *root = newNode(12);
    root->left = newNode(5);
    root->left->left = newNode(3);
    root->left->right = newNode(9);
    root->left->left->left = newNode(1);
    root->left->right->left = newNode(1);
    if (check(root))
        printf("Leaves are at same level\n");
    else
        printf("Leaves are not at same level\n");
    getchar();
    return 0;
}
```

Java

```
// Java program to check if all leaves are at same level

// A binary tree node
class Node
{
    int data;
    Node left, right;

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

class Leaf
{
    int leaflevel=0;
}

class BinaryTree
{
    Node root;
    Leaf mylevel = new Leaf();

    /* Recursive function which checks whether all leaves are at same
       level */
    boolean checkUtil(Node node, int level, Leaf leafLevel)
    {
        // Base case
        if (node == null)
            return true;

        // If a leaf node is encountered
        if (node.left == null && node.right == null)
        {
            // When a leaf node is found first time
            if (leafLevel.leaflevel == 0)
            {
                // Set first found leaf's level
                leafLevel.leaflevel = level;
                return true;
            }

            // If this is not first leaf node, compare its level with
            // first leaf's level
            return (level == leafLevel.leaflevel);
        }

        // If this node is not leaf, recursively check left and right
        // subtrees
        return checkUtil(node.left, level + 1, leafLevel)
                && checkUtil(node.right, level + 1, leafLevel);
    }

    /* The main function to check if all leafs are at same level.
       It mainly uses checkUtil() */
    boolean check(Node node)
    {
        int level = 0;
        return checkUtil(node, level, mylevel);
    }

    public static void main(String args[])
    {
        // Let us create the tree as shown in the example
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(12);
        tree.root.left = new Node(5);
```

```
        tree.root.left.left = new Node(3);
        tree.root.left.right = new Node(9);
        tree.root.left.left.left = new Node(1);
        tree.root.left.right.left = new Node(1);
        if (tree.check(tree.root))
            System.out.println("Leaves are at same level");
        else
            System.out.println("Leaves are not at same level");
    }
}

// This code has been contributed by Mayank Jaiswal
```

Python

```
# Python program to check if all leaves are at same level

# A binary tree node
class Node:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# Recursive function which check whether all leaves are at
# same level
def checkUtil(root, level):

    # Base Case
    if root is None:
        return True

    # If a tree node is encountered
    if root.left is None and root.right is None:

        # When a leaf node is found first time
        if check.leafLevel == 0 :
            check.leafLevel = level # Set first leaf found
            return True

        # If this is not first leaf node, compare its level
        # with first leaf's level
        return level == check.leafLevel

    # If this is not first leaf node, compare its level
    # with first leaf's level
    return (checkUtil(root.left, level+1)and
            checkUtil(root.right, level+1))

def check(root):
    level = 0
    check.leafLevel = 0
    return (checkUtil(root, level))

# Driver program to test above function
root = Node(12)
root.left = Node(5)
root.left.left = Node(3)
root.left.right = Node(9)
root.left.left.left = Node(1)
root.left.right.left = Node(2)

if(check(root)):
    print "Leaves are at same level"
else:
    print "Leaves are not at same level"

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

Output:

```
Leaves are at same level
```

Time Complexity: The function does a simple traversal of the tree, so the complexity is O(n).