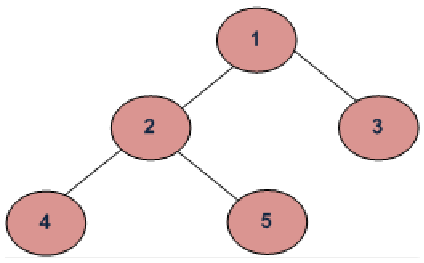
Program to count leaf nodes in a binary tree

A node is a leaf node if both left and right child nodes of it are NULL.

Here is an algorithm to get the leaf node count.

```
getLeafCount(node)
1) If node is NULL then return 0.
2) Else If left and right child nodes are NULL return 1.
3) Else recursively calculate leaf count of the tree using below formula.
    Leaf count of a tree = Leaf count of left subtree +
                                 Leaf count of right subtree
```



Example Tree

Leaf count for the above tree is 3.

Implementation:

C

```
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Function to get the count of leaf nodes in a binary tree*/
unsigned int getLeafCount(struct node* node)
{
  if(node == NULL)
    return 0;
  if(node->left == NULL && node->right==NULL)
    return 1;
  else
    return getLeafCount(node->left)+
           getLeafCount(node->right);
}

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
  struct node* node = (struct node*)
                       malloc(sizeof(struct node));
  node->data = data;
  node->left = NULL;
  node->right = NULL;

  return(node);
}

/*Driver program to test above functions*/
int main()
{
  /*create a tree*/
  struct node *root = newNode(1);
  root->left        = newNode(2);
  root->right       = newNode(3);
  root->left->left  = newNode(4);
  root->left->right = newNode(5);

  /*get leaf count of the above created tree*/
  printf("Leaf count of the tree is %d", getLeafCount(root));

  getchar();
  return 0;
}
```

Java

```java
//Java implementation to find leaf count of a given Binary tree

/* Class containing left and right child of current
 node and key value*/
class Node
{
    int data;
    Node left, right;

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

public class BinaryTree
{
    //Root of the Binary Tree
    Node root;

    /* Function to get the count of leaf nodes in a binary tree*/
    int getLeafCount()
    {
        return getLeafCount(root);
    }

    int getLeafCount(Node node)
    {
        if (node == null)
            return 0;
        if (node.left == null && node.right == null)
            return 1;
        else
            return getLeafCount(node.left) + getLeafCount(node.right);
    }

    /* Driver program to test above functions */
    public static void main(String args[])
    {
        /* create a tree */
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);

        /* get leaf count of the abve tree */
        System.out.println("The leaf count of binary tree is : "
                             + tree.getLeafCount());
    }
}

// This code has been contributed by Mayank Jaiswal(mayank_24)
```

Time & Space Complexities: Since this program is similar to traversal of tree, time and space complexities will be same as Tree traversal (Please see our Tree Traversal post for details)