**Problem-1:**

**Statement:** The binary tree data structure has two attributes left and right, indicating the left and right

subtrees. If a node has no left (or right) subtree, left (or right) is NIL. The depth of a binary

tree is the longest number of steps (distance) from the root to a leaf.

Use mathematical induction to explain that the Depth algorithm correctly computes the

depth of a binary tree.

DEPTH(T)

1. if T == NIL then

2. return -1

3. if T.left == NIL and T.right ­== NIL then

4. return 0

5. return 1 + max( Depth(T.left), Depth(T.right) )

**Solution:**

We will prove by mathematical induction that; Depth algorithm correctly computes the depth of a binary tree, for all input sizes (number of nodes of a tree).

**Step -1:**

If binary tree has 0 node. The longest number of steps from root to leaf is undefined. Depth returns -1 to indicate undefined depth. **[From line#1,2]**

If binary tree has 1 node. The longest number of steps from root to lead is 0. As, root and leaf is same. Depth returns 0 in this case. **[From line#3,4]**

Therefore, Depth correctly computes the depth of a binary tree for minimum input size (number of node).

So, Step 1 is proved.

**Step -2:**

We have to prove that, if Depth computes correctly the depth of a binary tree T for input sizes (number of nodes) 0,1,2,3,4,5,…, m; then it will compute correctly the depth of the binary tree T for input size (number of nodes) m + 1.

Assume, Depth computes correctly the depth of a binary tree for input size(number of nodes) 0,1,2,3,…, m.

Let say, T has m + 1 nodes.

Therefore, number of nodes for T.left or T.right <= m + 1 – 1 = m

As we assume that, Depth computes correctly the depth of T for number of nodes 0,1,2,3,…, m.

So, Depth computes correctly depth of T.left and T.right binary subtree. **[According to our assumption]**

***“max( Depth(T.left), Depth(T.right) )”*** computes the maximum depth between the two sub tree. It means it gives largest number of steps from a node(**root** for the **corresponding subtree**) to leaf node. The number step from that node to root is 1. So, ***“1 + max( Depth(T.left), Depth(T.right) )”*** - gives the maximum number of steps from root to leaf node. i.e. Depth correctly computes the depth of binary tree T for input size (number of nodes) m + 1.

Therefore, Step 2 is proved.

So, Depth computes correctly the depth of a binary tree T for all input sizes. **[Proved by mathematical induction]**

**Problem-2:**

**Statement:** A palindrome is a string that is the same as its reverse. For example, “abba” is a palindrome.

Use mathematical induction to prove that the algorithm Palindrome correctly determines if

an input string s is a palindrome.

PALINDROME(s):

1. if len(s) <= 1 then

2. return True

3. return s[0] == s[len(s) - 1] and PALINDROME(s[1: len(s) - 1])

**Solution:**

We will prove by mathematical induction that; the algorithm PALINDROME correctly determines if an input string s is a palindrome. i.e. This will work correctly for all length of s.

**Step – 1:**

A string of length 1 or 0 is palindrome.

From line 1 and 2, PALINDROME returns true for string length of 1 or less.

So, step 1 is proved. i.e. PALINDROME works correctly for minimum input size.

**Step – 2:**

We have to prove that, if PALINDROM determines correctly if an input string s is a palindrome, for input size (length of s): 0,1,2,3,4, …m. Then it will determine correctly if an input string s is a palindrome, for input size (length of s): m + 1.

Assume, PALINDROME works correctly for input size (length of s) 0,1,2,3,4, …m – 1, m.

Let say, s is a string of length m + 1.

From line 3, PALINDROME determines input string s as palindrome, **if**

* 1st and m+1th (last element) of string are equal **[statement 1]**

**AND**

* PALINDROME defines the string from 2nd to mth as palindrome. **[statement 2]**

**Otherwise**, PALINDROME will not determine the string as palindrome.

Here, if a string is palindrome then its first and last element will be same. So, **statement 1** will be true for a palindrome.

And, the string between first and last element of palindrome is also a palindrome. As, reverse of a palindrome string is same. Here, As we assume that PALINDROME works correctly for string of length 0, 1,2, …, m-1,m. And, The length of the string from 2nd to mth is m – 1. Therefore, **statement 2** will be true for a palindrome string.

Therefore, PALINDROME will correctly determine if an input string is palindrome, for input size (length of s) m + 1.

So, **step 2** is proved.

Therefore, **PALINDROME** correctly determines if an input s is palindrome for all input sizes (length of s). **[proved by mathematical induction]**