Daily Coding Problem

Good morning! Here's your coding interview problem for today.

This problem was asked by Google.

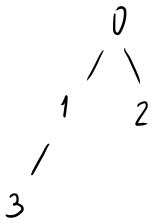
Given the root to a binary tree, implement `serialize(root)`, which serializes the tree into a string, and `deserialize(s)`, which deserializes the string back into the tree.

For example, given the following `Node` class

```
class Node:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

The following test should pass:

```
node = Node('root', Node('left', Node('left.left')),
Node('right'))
assert deserialize(serialize(node)).left.left.val ==
'left.left'
```



funct trav(Node):
if (no leaves) return Node;
trav(Node.left);
trav(Node.righ);

Object oriented programming: C++

class C {
C i; ← illegal, if every object C must contain C, it is ∞... is what I read but why not assume NULL? is it b/c C must allocate memory upon initialization?
};

Anyways: class C { C* i; } OR class C { C& i; };

default (seems to be, in tutorials etc.)

what would be advantage of this?

I'm running into problems w/ Constructors and in traversal

Goal. $C(x_0, C(x_1, C(x_2, C(x_3)))$; ← valid

C_0.next.next.next.value returns x_3 ← valid.

this problem probably isn't worth the afford but:

constructor: C(x, &C) → "cannot bind lvalue to rvalue"
C(x, *C) → "
C(x, C) → overwrites pushed data values

Solutions

(1) Constructors:

when C's constructor needs an arg of type C, make it permanent by calling

```
C* temp = new C(arg...);
```

(2) Traversal:

use accessor functions to do so:

☑ C.next() ☒ C.next