Data Abstraction and Trees!

1st March, 2023

Outline For Todays Section!

Total Time: 50 mins

Introductions: 5 mins

Mini Lecture: 10 mins

Question 1: 5 mins

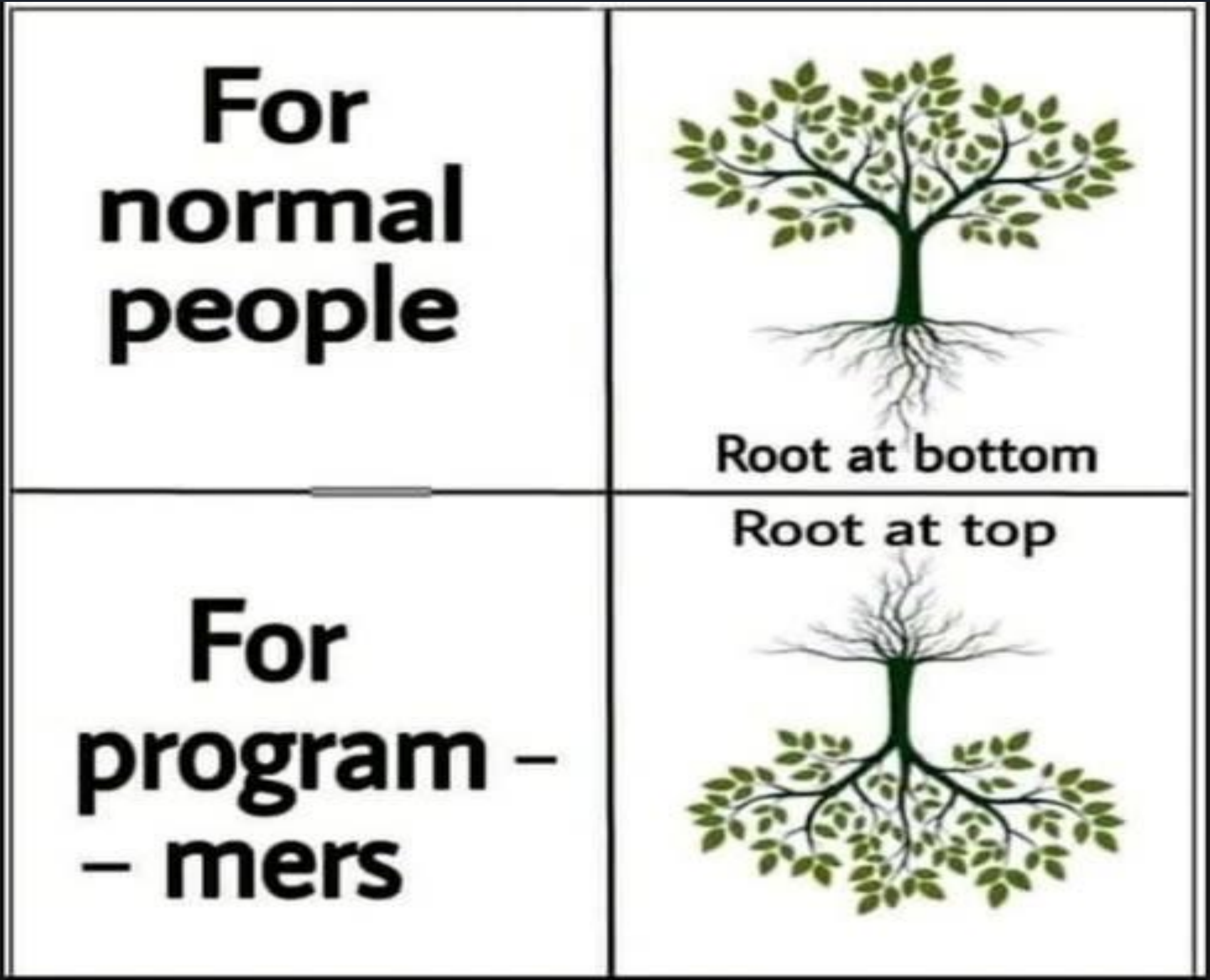
Question 2: 10 mins

Question 1 (Trees) : 10 mins

Question 2 (Trees) : 10mins

[Optional] Q&A: 15mins

[Optional] Exam Level Problem


For normal people
Root at bottom
Root at top
For program – – mers

Abstract Data Types (ADT)

For ADT’s we only care about the behaviour and not the abstraction!

To get an intuitive idea of what this means, let’s consider an example:

- When you are using integers in python, you just care about how integers behave and what you can do with them and not necessarily how the Integer class is implemented in python.
- An ADT is analogous to a black box which hides the inner implementation.

Constructors

- Constructors create abstract data types, and define traits and attributes that make that specific data type unique!
- Imagine we're in a car factory that can make cars of any make and model. We can construct cars through assigning a variable to an instance of the car constructor with specific parameters specified:
  - C = car("make", "model")
- We can create constructors by specifying variable attributes as arguments, and assignin
- Source: CSM Content Team

Selectors

- We use selectors to access and use data types (rather than directly accessing it from the constructor)
- This way we don’t break the abstraction barrier!

Abstraction Barriers

Trees!

STRUCTURE:

1. Root
2. Branches
3. Leaves

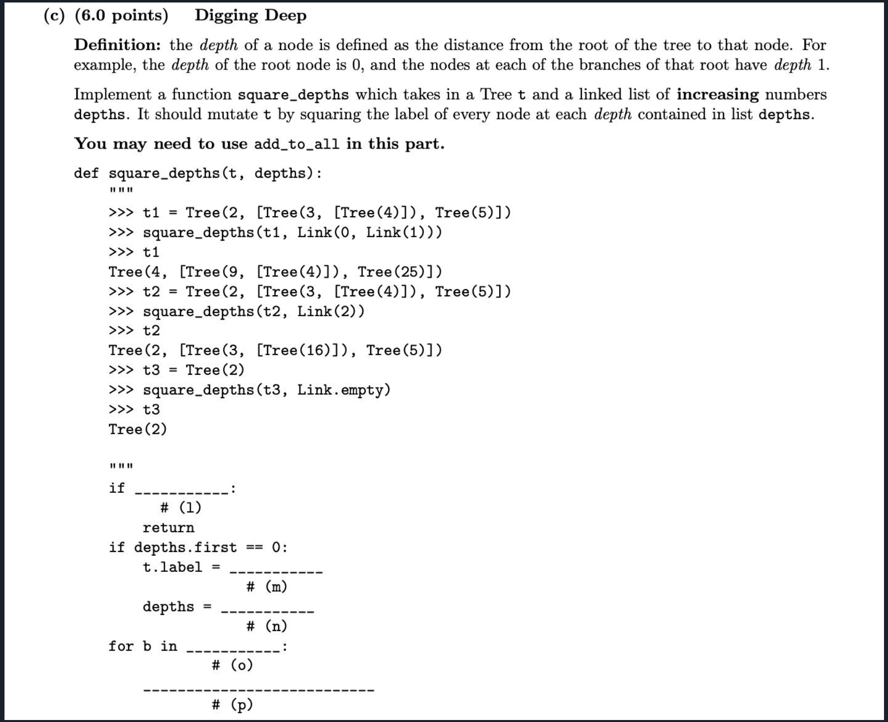
Trees are **recursive** data structure! (each branch is also a tree)

THERE ARE EXCEPTIONS TO THIS!!

General Approach for tree problems (Credits: CSM Content Team)

- The recursive nature of trees lend themselves to using recursion in one way or another to traverse the “layers” of a tree and change/”prune” nodes that fulfill a certain condition.
- When we recurse through trees…
    - Our base case is when we reach a leaf.
    - Our recursive case is traversing through the branches.
        - NOTE: In pretty much every tree problem, we always have a for-loop looping through ‘for b in branches ’ in which we apply our recursive method to each ‘b.’ This is because each ‘b’ is a subtree in itself that needs to be broken into its own base case.
            - The location of your for loop matters! When we want branches to be immediate results of our function on the parent nodes, we put the for loop in the end. When we want our parent nodes to be a cumulative of whatever function we have, we generally put the for loop in the beginning.

Exam Level Question (su21)



(c) (6.0 points) Digging Deep

Definition: the *depth* of a node is defined as the distance from the root of the tree to that node. For example, the *depth* of the root node is 0, and the nodes at each of the branches of that root have *depth* 1.

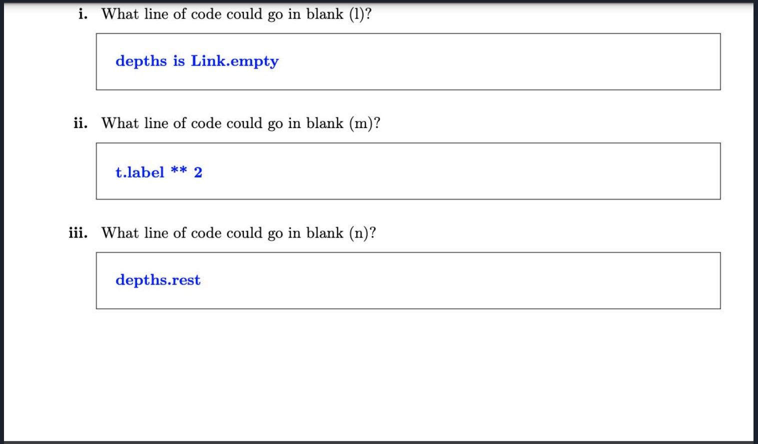
Implement a function `square_depths` which takes in a Tree `t` and a linked list of **increasing** numbers `depths`. It should mutate `t` by squaring the label of every node at each *depth* contained in list `depths`.

You may need to use `add_to_all` in this part.

```
def square_depths(t, depths):
    """
    >>> t1 = Tree(2, [Tree(3, [Tree(4)]), Tree(5)])
    >>> square_depths(t1, Link(0, Link(1)))
    >>> t1
    Tree(4, [Tree(9, [Tree(4)]), Tree(25)])
    >>> t2 = Tree(2, [Tree(3, [Tree(4)]), Tree(5)])
    >>> square_depths(t2, Link(2))
    >>> t2
    Tree(2, [Tree(3, [Tree(16)]), Tree(5)])
    >>> t3 = Tree(2)
    >>> square_depths(t3, Link.empty)
    >>> t3
    Tree(2)

    """
    if ___________:
          # (l)
        return
    if depths.first == 0:
        t.label = ___________
                     # (m)
        depths = ___________
                     # (n)
    for b in ___________:
                 # (o)
        ___________________________
                 # (p)
```

Exam Level Question Solution



i. What line of code could go in blank (l)?

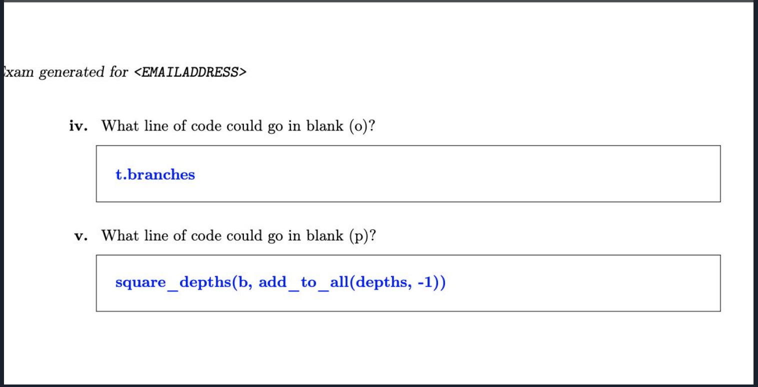
```
depths is Link.empty
```

ii. What line of code could go in blank (m)?

```
t.label ** 2
```

iii. What line of code could go in blank (n)?

```
depths.rest
```



xam generated for <EMAILADDRESS>

iv. What line of code could go in blank (o)?

```
t.branches
```

v. What line of code could go in blank (p)?

```
square_depths(b, add_to_all(depths, -1))
```

Anonymous Feedback Form