**Tree Abstraction**

**def** **tree**(root, branches=[]):

**return** [root] + list(subtrees)

**def** **root**(t):

**return** t[0]

**def** **branches**(t):

**return** t[1:]

**def** **is\_leaf**(t):

**return** **not** subtrees(t)

Remember, trees are **recursively defined** (trees are constructed using smaller trees). For most questions involving the tree ADT, you can break down the thought process into three steps:

1. **Base case**: Usually, this is if the tree is a leaf (use the is\_leaf function)
2. **Recursive call**: Consider what a recursive call on a single branch will do. What information does it give you?
3. **Recursive case**: Make recursive calls on each branch (using a for loop or a list comprehension) and combine that in some way with the root for your final answer.

Q1:

Complete the implementation of *num\_nodes*, which takes a tree *t* and returns the number of nodes in the tree rooted at *t*.

def num\_nodes(t):

""" Return the number of nodes in t.

>>> num\_nodes( tree(1) )

1

>>> num\_nodes( tree(1, [tree(2), tree(3)]) )

3

"""

if \_\_\_\_\_\_:

return 1

return \_\_\_ + sum(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Q2:

def print\_level\_order(t):

"""

>>>t=tree(1,[tree(3,[tree(2),tree(4)]),tree(6,[tree(5),tree(7)])])

>>>print\_level\_order(t)  
1

3

2

4

6

5

7 """

def helper(t):

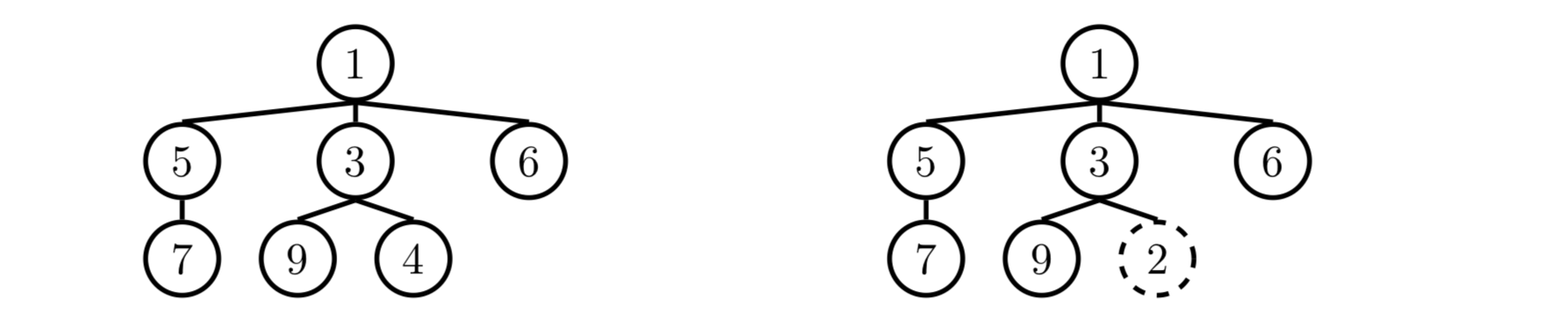
children=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[helper(\_\_\_\_\_)for\_\_\_\_\_in\_\_\_\_\_\_\_\_\_\_\_]

helper(t)

Q3:

A min-heap is a tree with the special property that every node’s value is less than or equal to the values of all of its branches. 

Implement **is\_min\_heap** which takes in a tree data abstraction and returns whether the tree satisfies the min-heap property or not.

Environment Diagrams Review:

1.

def bo(jack):

return lambda mask: horse

def horse(mask):

horse = mask

def mask(horse):

return horse

return horse(mask)

jack, mask = horse, bo

hollywoo = bo(jack)(horse)(mask)

2. Spring 2018 Environment Diagram

3. Fall 2016 Environment Diagram

HOF Code writing questions

1.

**def** memory(x, f):  
"""Return a higher-order function that prints its memories.

>>> f = memory(3, lambda x: x)

>>> f = f(square)

3

>>> f = f(double)

9  
>>> f = f(print) 6  
>>> f = f(square) 3  
None  
"""  
**def** g(h):

**print**(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**return** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**return** g

2. Spring 2018 Midterm 1 #4

3. Spring 2018 Final #5a