Take a deep breath.

This is your chance to show us

How much you’ve learned.

We WANT to give you the credit

That you’ve earned.

A number does not define you.

Good luck!
1. [5] Write a function `filter_pos_rec`, which takes a recursive linked list, and outputs a new recursive linked list containing the same items as the original, except with the ones less than or equal to 0 removed.

You may **not** use a loop, nor any `LinkedListRec` methods other than the constructor and `is_empty` in your solution. Instead, use recursion to access and set the attribute directly! (Remember, recursive code is pretty short: you can do this in under 6 lines!)

```python
1 def filter_pos_rec(lst):
2     """ (LinkedListRec of int) -> LinkedListRec of int
3
4     Return a new LinkedListRec whose items are
5     the ones in lst that have value > 0.
6     The items must appear in the *same order*
7     they do in lst.
8
9     >>> lst = LinkedListRec([3, -10, 4, 0])    # [3 -> -10 -> 4 -> 0]
10    >>> pos = filter_pos_rec(lst)              # pos is [3 -> 4]
11   """
```
2. (a) [2] Draw two **different** binary search trees which both contain the items 1, 3, 5, 8, 10, 11. Label them **BST1** and **BST2**.

(b) [2] Write the list returned from the **preorder traversal of BST1**, and then the list returned from the **postorder traversal of BST2**. Clearly state which one is which!

Note that you will receive a *zero* for this question if you haven’t labelled your trees in part (a)!

(c) [3] A mystery binary tree (NOT a binary search tree) has preorder traversal [4, 3, 7, 1, 10, 6] and inorder traversal [7, 3, 10, 1, 4, 6]. Draw this binary tree. (*Warning*: this question is a little tricky; if you get stuck, move on!)
3. Recall that the **depth** of a node in a tree is its distance from the root (where the root has depth 1). For example, the tree below has five nodes at depth 3:

![Tree Diagram]

In this question, you will write a recursive function to determine the number of nodes in a tree of a certain depth.

(a) [1] State, in English, the relationship between the number of nodes of a tree at depth \( d \) with the nodes at depth \( d - 1 \) in its subtrees.

(b) [4] Implement the function below using recursion. You may use a loop to process all of the trees in the `subtrees` attribute, but you may not use any `Tree` methods (other than `count_depth`, obviously!).

```python
1  def count_depth(tree, d):
2      """ (Tree, int) -> int
3      Return the number of nodes in tree at depth d.
4      You may assume that d >= 1.
5      """
```
4. Here is a question regarding the `BinarySearchTree` class. Assume that we have implemented a method `size` for this class, which returns the number of items contained in a BST.

(a) [1] Suppose we are searching for the 10th smallest element in a BST, and suppose we know that the left subtree has 3 nodes. Should we next search in the left or right subtree, and what item should we be searching for in that subtree? (Hint: the answer “look for the 5th smallest item in the left subtree” is incorrect, but has the right structure for the answer.)

(b) [5] Your task is to implement the following function. You must use recursion, and may not use loops, or any other `BinarySearchTree` methods other than `size` and `is_empty`.

```python
def size(self):
    """ (BinarySearchTree) -> int
    Return the number of items contained in this BST.
    """
    ...
    # You are NOT required to implement this!

def kth_smallest(self, k):
    """ (BinarySearchTree, int) -> object
    Precondition: 1 <= k <= size of this BST
    Return the kth_smallest value in this BST. So if a BST b contains
    the items {1,3,5,8,10,11}, then b.kth_smallest(2) returns 3, and
    b.kth_smallest(5) returns 10.
    Note: the return value depends only on the items in the tree, and
    not the structure. So if two trees contain the same items, but have
    different roots, calling kth_smallest still returns the same value.
    """
    # YOUR CODE GOES HERE
```
Bonus Question [2]

Warning: this is a difficult question, and will be marked harshly. Only attempt it if you have finished all of the other questions!

Write a function pre_loop that takes a BST, and returns a preorder traversal of that BST, without using recursion.
Use this page for rough work.
Name:

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Total</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grade</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out Of</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>23</td>
<td>2</td>
</tr>
</tbody>
</table>