CSC148, Lab #8
week of November 4th, 2013

This document contains the instructions for lab number 8 in CSC148. To earn your lab mark, you must actively participate in the lab. We mark you in order to ensure a serious attempt at learning, **not** to make careful critical judgments on the results of your work.

general rules

We will use the same general rules as for the first lab (including pair programming). See the instructions at the beginning of Lab 1 to refresh your memory.

overview

In this lab, you will write two functions that combine working with trees and working with linked lists.

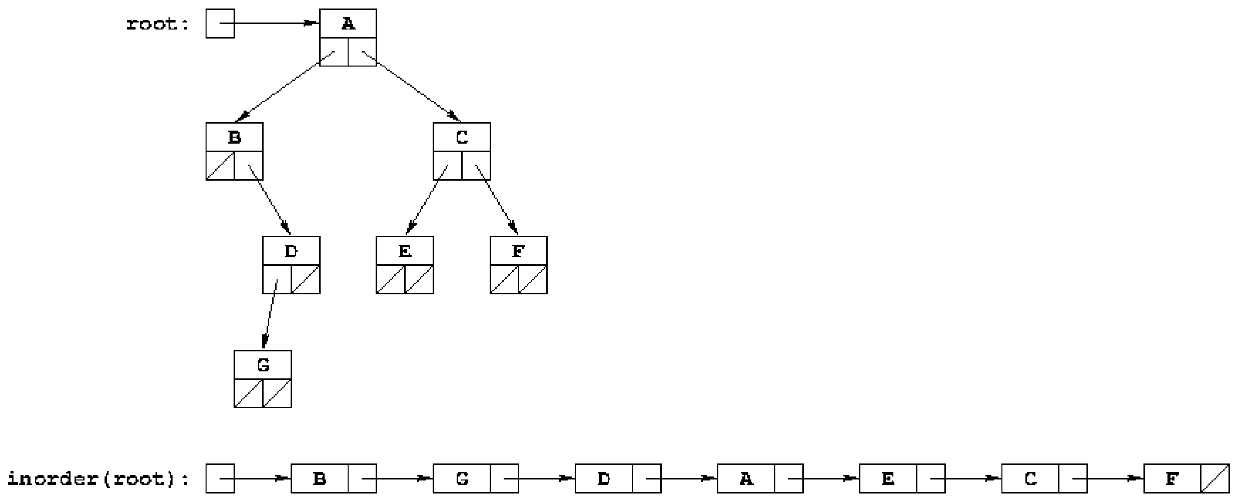
The code you have to write is really quite short and simple, if you see things a particular way. So the main goal of this lab is to get you to spend a significant amount of time thinking before your code. Draw lots of pictures, write down high-level outlines of the functions you have to write, make sure you spend a lot of time discussing how to use recursion to accomplish your goals. And in the end, with the help of your partner, other students in your lab, and your TA, you should be able to write the two functions below in no more than about a dozen lines of code (each)!

inorder traversal

- Agree on who will be student `s1` and who will be student `s2` for this lab. For this part, student `s1` drives and student `s2` navigates.
- In a file named `inorder.py`, write a `binary tree node` class and a separate `linked list node` class (yes, it's fine to just re-use classes we've used before).
- Then, write code for a function named `inorder` that takes the root node of a binary tree as an argument and that returns the first and last nodes in a linked list that contains every value from the binary tree, listed according to an inorder traversal:

```
def inorder(root: BTNode) -> (LLNode, LLNode):
    """Return the first and last nodes in a linked list that contains every
    value from the binary tree rooted at root, listed according to an inorder
    traversal.
    """
```

For example, if root is a reference to the root of the binary tree in the picture below, then your function should return a reference to the first and last nodes in the linked list pictured below the tree.



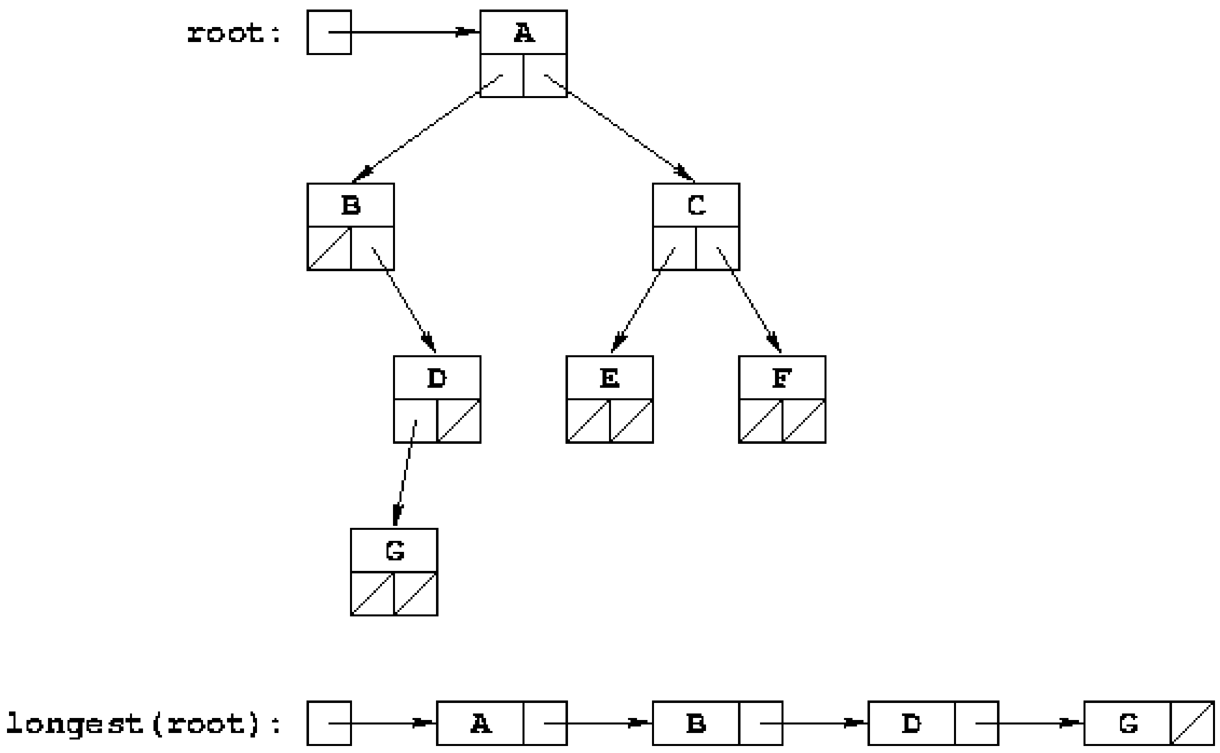
- **Note 1**: Your function must create all of the nodes in the linked list and link them together in the right way, making the links as it visits the corresponding nodes in the binary tree. You are not allowed to use a python `list` or other container class.
- **Note 2**: Think carefully about this function: when you're working on the root of some tree, what information would be most useful to get back from the recursive calls on the left and right sub-trees, to make it easy to put together the final answer? Write your code based on getting this information from each recursive call, and just make sure to return the same information back.
- When you are done, you may want to take a few minutes to put together some testing code — a simple loop to print all of the values in the linked list, for example.

When you're done, show your work to your TA and switch roles.

longest path

- For this part, student `s2` drives and student `s1` navigates.
- In a file named `longest.py`, import your `binary tree` node class and your `linked list` node class from `inorder`.
- Then, write code for a function named `longest` that takes the root of a binary tree as an argument and that returns the first node in a linked list that contains every value in a longest path from the root of the binary tree to one of its leaves.

  For example, if root is a reference to the root of the binary tree in the picture below, then your function should return a reference to the first node in the linked list pictured below the tree.



- **Note 1**: Your function must create all of the nodes in the linked list and link them together in the right way. You should create the link in the linked list at the same time as you are visiting the corresponding node in the binary tree — you are not allowed to use a python `list` or other container class.
- **Note 2**: You will find it much easier to write a recursive helper function and simply call it within the body of longest. Think carefully about this helper function: when you're working on the root of some tree, what information would be most useful to get back from the recursive calls on the left and right sub-trees, to make it easy to put together the final answer? Write your code based on getting this information from each recursive call, and just make sure to return the same information.
- When you are done, you may want to take a few minutes to put together some testing code — a simple loop to print all of the values in the linked list, for example.

When you are done, show your work to your TA. Then, please stick around to help other students in your lab section!