Exam 2 Practice / Lab 8

Data Structures Fall 2016

Schedule:

- First hour: closed book and notes. No collaboration.
- Second hour: open book, notes, and computer. Collaboration encouraged! Try your code out on your computer to see if it works.
- Third hour: TAs will go over the answers.

1. Consider the `PositionalList` class from the book. Write a method, `odd_iter`, that produces an iterator that iterates through the 1st, 3rd, 5th, etc., items in the list. You can not use any existing iterator methods directly.

   ```
   L=PositionalList()
   for i in range(10):
       L.add_last(i)
   for i in L.odd_iter():
       print(i)
   ```

   Should output 0, 2, 4, 6, 8.

2. Consider the `PositionalList` class from the book. Write a method, `swap_forward`, that takes a position and swaps the node and the one after it. To make this question more difficult, you are not allowed to use the `._element` field of the node class.

3. Suppose you wanted to insert the integers 1..7 into a binary search tree so that it would have as large a height as possible and you wanted to insert 3 last. Which order could you insert them?

4. Suppose you wanted to insert the integers 1..7 into a binary search tree so that it would have as small a height as possible. Which order would you insert them?

5. Write a method of `LinkedBinaryTree` called `number_at_depth(self,d)` that returns the number of nodes in the tree at depth *d*. You can not use any of the methods of `LinkedBinaryTree`. Just use the fact that it has a `._root`, and that the `LinkedBinaryTree._Node` class has `._element`, `._parent`, `._left` and `._right`

6. Given a string *s*, write a function `stringTree` to construct a tree (a `LinkedBinaryTree`) as follows: the root contains *s*, and the left and right children contain the first and second halves of *s*. This continues recursively and the leaves contain a single character.

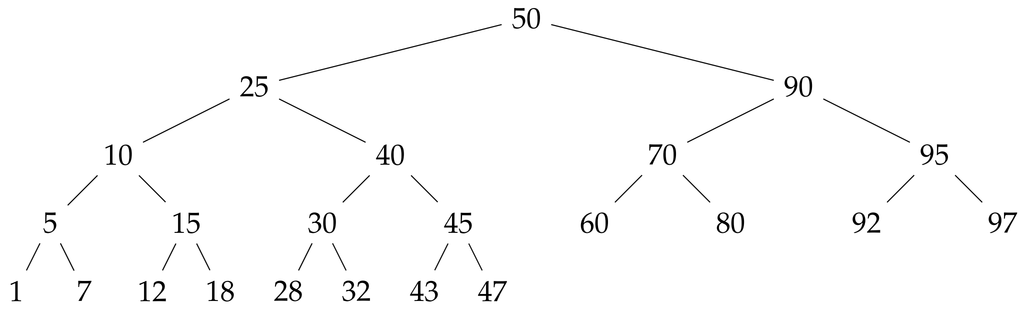
```
#Test code
T=stringTree("Hello!")
preorder_indent(T,T.root(),0)
##Hello!
##  Hel
##    H
##    el
##      e
##      l
##  lo!
##    l
##    o!
##      o
##      !
```

7. Consider the `PositionalList` class from the book. Write a method `location(self,p)` which returns a integer *i* if *p* represents the *i*th item from the front of the list. All you need to know to answer this question is that a position has `._container` and `._node`; `PositionalList` has `._header` and `._trailer`; and that `PositionalList._Node` has `._element`, `._prev` and `._next`.

   ```
   L=PositionalList()
   for i in range(10):
       L.add_last("donkey")
   print(L.location(P.first()))
   print(L.location(P.last()))
   print(L.location(L.before(P.last())))
   ```

   Should output 1, 10 and 9

8. Consider the following binary tree:



   (i) Is this a binary search tree? Answer yes or no. If your answer is no, draw on the tree showing the simplest way to make it a binary search tree.

   (ii) What is LAST node in the inorder traversal of this tree?

   (iii) What is LAST node in the preorder traversal of this tree?

(iv) What is LAST node in the postorder traversal of this tree?

(v) What is LAST node in the level-order (breath-first) traversal of this tree?

(vi) Show how to insert 11 into this tree, writing "I" next to your answer.

(vii) What would you delete that will result in moving 28? (write your answer here, do not answer 28). Show on the above digram how to do it, writing "D" next to your answer.

9. Suppose we wanted to change the `_Node` class of `LinkedBinaryTree` to add two more fields, `_next` and `_prev` which point to the next and previous nodes in the tree in an inorder traversal. Assuming that all other methods have been coded correctly to handle this, your job is to code `_add_left`. Here is the current code:

```
def _add_left(self, p, e):
  node = self._validate(p)
  if node._left is not None:
    raise ValueError('Left child exists')
  self._size += 1
  node._left = self._Node(e, node)                  # node is its parent
  return self._make_position(node._left)
```

Note that you will need to change the call to the `_Node` constructor, which will now take two additional parameters, but there are other additions you will need to make as well.