Worksheet 7 Solutions

```
def second_to_last(t):
    """Prune the second to last node in the tree. Do not prune the root.
        >>> t = Tree(1, [Tree(2), Tree(3, [Tree(4), Tree(5), Tree(6,
[Tree(7)])])])
        >>> double_wubble(t)
        >>> t
        Tree(1, [Tree(2), Tree(4), Tree(5), Tree(7)])
        """
    for b in t.branches:
        for b2 in b.branches:
            if b2.is_leaf() :
                keep = b.branches
                t.branches.remove(b)
                t.branches.extend(keep)
                break
        second_to_last(b)
```

```
def backwards_forwards(l):
    """Make the linked lists a backwards version of itself. Remember, this is
        a mutable function. Assume the linked list is not circular.
        >>> l = Link(1, Link(2, Link(3)))
        >>> backwards_forwards(l)
        >>> l
        Link(3, Link(2, Link(1)))
        >>> l2 = Link(1, Link(Link(2, Link(3)), Link(4, Link(5))))
        >>> backwards_forwards(l2)
        >>> l2
        Link(5, Link(4, Link(Link(2, Link(3)), Link(1))))
        """
    beg, end = l, l
    while (end.rest != Link.empty):
        end = end.rest
    while (beg is not end and end.rest is not beg):
        end.first, beg.first = beg.first, end.first
        beg = beg.rest
        temp, end = end, l
        while (end.rest is not temp):
            end = end.rest
```

Θ(n)
Θ(n)
Θ(log(n))
Θ(n^2)