Announcements

- Midterm 2 is on Monday 10/28 7pm-9pm
  - Topics and locations: [http://inst.eecs.berkeley.edu/~cs61a/fa13/exams/midterm2.html](http://inst.eecs.berkeley.edu/~cs61a/fa13/exams/midterm2.html)
  - Bring 1 hand-written, 2-sided sheet of notes. Two study guides will be provided.
  - Emphasis: mutable data, object-oriented programming, recursion, and recursive data
  - Have an unavoidable conflict? Fill out the conflict form by Friday 10/25 @ 11:59pm!
- Review session on Saturday 10/26 from 1pm to 4pm in 1 Pimentel
- HKN review session on Sunday 10/27 from 4pm to 7pm in 2050 VLSB
  - Includes content through Wednesday 10/23 (today is review & examples)
- No lab next Monday, Tuesday, & Wednesday
- Homework 7 is due Tuesday 11/5 @ 11:59pm (Two weeks)

Recursive Lists Can Change

Attribute assignment statements can change first and rest attributes of an Rlist.
The rest of a recursive list can contain the recursive list as a sub-list.

```python
>>> s = Rlist(1, Rlist(2, Rlist(3)))
>>> t = s.rest
>>> t.rest = s
>>> s.first
5
>>> s.rest.rest.rest.first
2
```

Note: The actual environment diagram is much more complicated.

Mutable Recursive Lists

Recursive Lists as Functions

The object system is convenient, but it isn’t necessary for designing data types!
Write a function `pruned` that takes two `Tree` arguments `t1` and `t2` and returns whether `t2` is a pruned version of `t1`. `t2` is a pruned version of `t1` if all paths from the root of `t2` are also valid paths from the root of `t1`.

### Pruned Tree Examples

<table>
<thead>
<tr>
<th>(a,b)</th>
<th>(a,c)</th>
<th>(a,d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

### Recursive Idea

- `pruned(a, c)` implies
- `pruned(a.right, c.right)`

- What about `c.left`?

### Recursive Implementation

```python
def pruned(t1, t2):
    if t2 is None:
        return True
    elif t1 is None:
        return False
    else:
        return pruned(t1.left, t2.left) and pruned(t1.right, t2.right)
```
Go Bears!

```python
def oski(bear):
    def cal(berk):
        nonlocal bear
        if bear(berk) == 0:
            return (berk+1, berk-1)
        bear = lambda ley: berk-ley
        return (berk, cal(berk))
    return cal(2)
oski(abs)
```

Non-Local Assignment Variants

Go Bears!

```python
def oski(bear):
    def cal(berk):
        nonlocal bear
        if bear(berk) == 0:
            return (berk+1, berk-1)
        bear = lambda ley: berk-ley
        return (berk, cal(berk))
    return cal(2)
oski(abs)
```