Monday, March 16

Announcements

- Midterm 2 is on Thursday 3/19 7pm-9pm
- Topics and locations: http://cs61a.org/exams/midterm2.html
- Bring 1 hand-written, 2-sided sheet of notes; 2 study guides will be provided
- Emphasis: mutable data, object-oriented programming, recursion, and recursive data
- Review session on Tuesday 5:00pm-6:30pm in 2050 VLSB
- Includes content through Friday 3/13 (today is review & examples)

- No lecture next Wednesday 3/18
- No discussion sections next Thursday 3/19 or Friday 3/20
- Lecture next Friday 3/20 is a video (but a great one)

Linked Lists

```python
s = Link(1, Link(2, Link(3)))
```

```python
s.first = 5
```

```python
t = s.rest
```

```python
t.rest = s
```

Recursive Lists Can Change

Attribute assignment statements can change first and rest attributes of a Link.
The rest of a linked list can contain the linked list as a sub-list.

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)
```

```
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)
```

```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
Morse Code

Morse code is a signaling protocol that transmits messages by sequences of signals.

A: ● ● ●
B: ● ● ● ● ● ●
C: ● ● ● ● ●
D: ● ● ● ● ● ● ●
E: ●
...

Problem: Implement `morse` so that `decode` works correctly.

```python
def morse(code):
    ...

abcde = ('A': '-..', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.')
def decode(signals, tree):
    """Decode signals into a letter using a morse code tree."
    
    tree = morse(abcde)
    for signal in signals:
        if signal == '.':
            tree = tree.left
        elif signal == '-':
            tree = tree.right
    return tree.entry
```

(See the code for a complete implementation.)