Linked Lists

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

```
>>> s = Link[1, Link[2, Link[3]]]
>>> s.first = 5
>>> t = s.rest
>>> s.first
>>> t = s.rest.rest.rest.first
2
```

Note: The actual environment diagram is much more complicated.

Environment Diagrams

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

```
>>> s = Link[1, Link[2, Link[3]]]
>>> s.first = 5
>>> t = s.rest
>>> s.first
>>> t = s.rest.rest.rest.first
2
```

Note: The actual environment diagram is much more complicated.

Go Bears!

```
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)
return oski(abs)
```

```
class Worker:
    greeting = 'Sir'
    def __init__(self):
        self.elf = Worker
    def work(self):
        return self.greeting + ', I work'
    def __repr__(self):
        return 'Bourgeoisie.greeting: ' + "'Sir'"

class Bourgeoisie(Worker):
    greeting = 'Peon'
    def work(self):
        print(Worker.work(self))
        return 'I gather wealth'
    def __repr__(self):
        return "Bourgeoisie.greeting: 'Peon'"

jack = Worker()
john = Bourgeoisie()
jack.greeting = 'Maam'
```
Morse Code

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

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

```python
abode = ('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z')
def decode(signals, tree):
    """Decode signals into a letter."
    for signal in signals:
        tree = [b for b in tree.branches if b.entry == signal][0] # An empty list is a false value
    leaves = [b for b in tree.branches if not b.branches]
    assert len(leaves) == 1
    return leaves[0].entry
```

```
A: ● ● ● ● ●
B: ● ● ● ●
C: ● ● ●
D: ● ● ●
E: ● ● ●

abcde = ('a', 'b', 'c', 'd', 'e')
t = morse(abode)

[decode(s, t) for s in ['-..', '.', '-.-.', '.-', '-..', '.']]

['d', 'e', 'c', 'a', 'd', 'e']
```

Not used '.

'a'

'-'...

An empty list is a false value