61A Lecture 23

Friday, October 24

Announcements

- Midterm 2 is on Monday 10/27 7pm–9pm

  - Topics and locations: http://cs61a.org/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

  - Review session on Saturday 10/25 from 3pm to 6pm in 2050 VLSB

  - HKN review session on Sunday, 10/26 12–3pm in 155 Dwinelle

  - Includes content through Wednesday 10/22 (today is review & examples)

  - Discussion handouts contain practice questions!

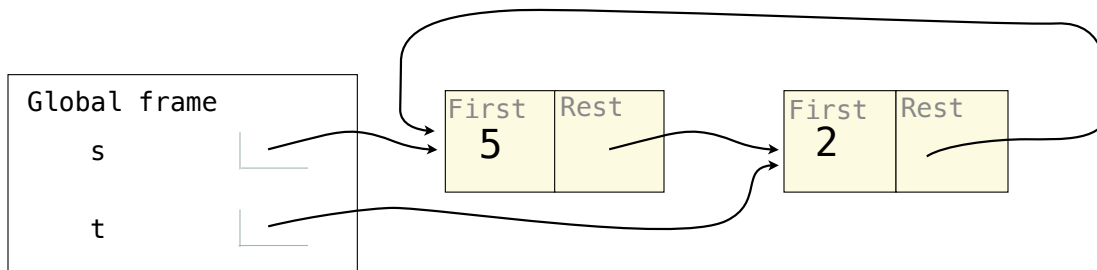- No lecture next Monday, No lab next Tuesday & Wednesday, No office hours next Mon–Wed
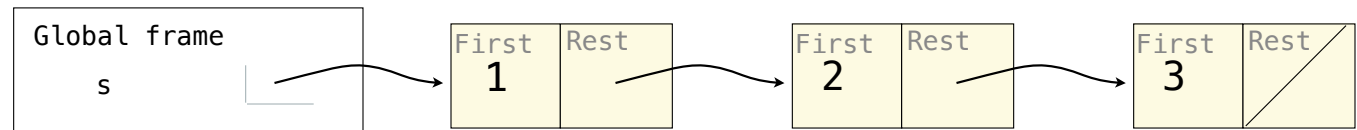
Mutable Linked Lists

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
>>> t.rest = s
>>> s.first
5
>>> s.rest.rest.rest.rest.rest.first
2
```


Note: The actual environment diagram is much more complicated.

Hailstone Trees

Hailstone Trees

Pick a positive integer n as the start

If n is even, divide it by 2

If n is odd, multiply it by 3 and add 1

Continue this process until n is 1

Question: Write hailstone_tree(k, n=1),
which returns a Tree in which the paths
from the leaves to the root are all
possible hailstone sequences of length k
ending in n.

All possible n that start a
length–8 hailstone sequence ▶

```
1
|
2
|
4
|
8
|
16
|    \
32      5
|       |
64     10
|  \    |  \
128  21 20   3
```
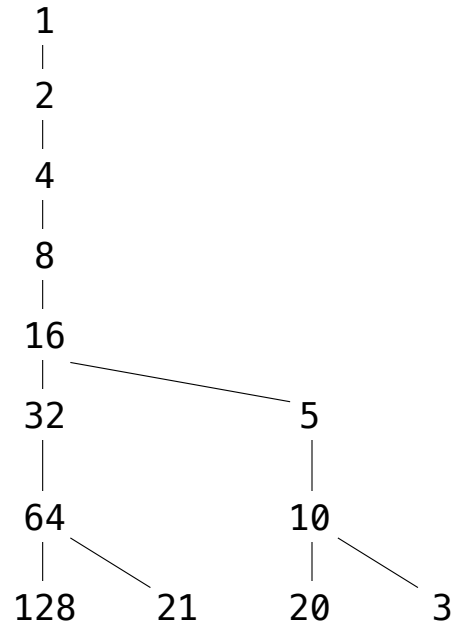
(Demo)

Question: List the leaves of a Tree instance
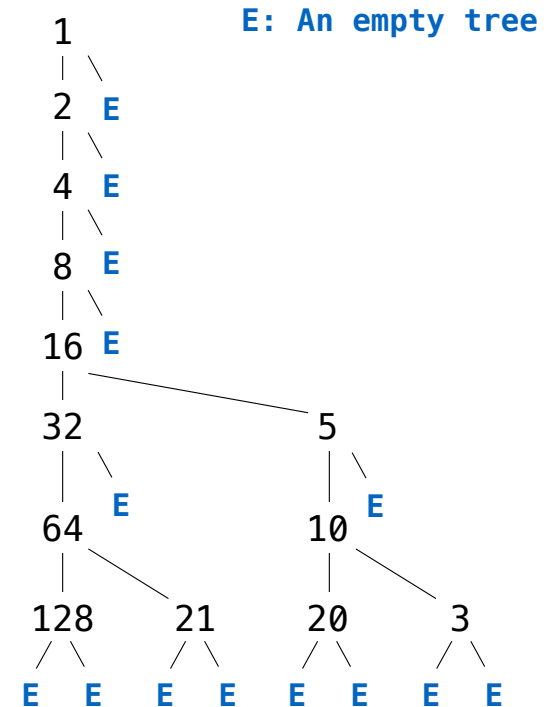
Binary Hailstone Trees

Hailstone Trees using BinaryTree

```python
class BinaryTree(Tree):
    empty = Tree(None)
    empty.is_empty = True

    def __init__(self, entry, left=empty, right=empty):
        Tree.__init__(self, entry, (left, right))
        self.is_empty = False

    @property
    def left(self):
        return self.branches[0]

    @property
    def right(self):
        return self.branches[1]
```

E: An empty tree

```
 1
 | \
 2  E
 |  \
 4   E
 |   \
 8    E
 |     \
16      E
 |
32              5
 |   \          | \
64    E        10   E
 |    \         |   \
128   21       20    3
/ \   / \      / \   / \
E E   E E      E E   E E
```

(Demo)

Question: List the entries in the longest path in a binary tree
for which all elements are less than k

Environment Diagrams

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)

oski(abs)
```

Global frame

oski → func oski(bear)[parent=G]

f1: oski [parent=G]

bear → ✗ → func λ(ley) [parent=f2]

→ func abs(...) [parent=G]

cal → func cal(berk) [parent=f1]

Return Value

f2: cal [parent=f1]

▶ berk 2

Return Value

list
0	1
2	

f3: cal [parent=f1]

berk 2

Return Value

list
0	1
3	1

f4: λ [parent=f2]

ley 2

Return Value 0

[2, [3, 1]]