Hog Contest Rules

• Up to two people submit one entry;
• Max of one entry per person
• Your score is the number of entries against which you win more than 50.00001% of the time
• One more rule! Pork Chop
• All strategies must be deterministic, pure functions of the current player scores
• All winning entries will receive 2 points of extra credit
• The real prize: honor and glory

Fall 2011 Winners
Kaylee Mann
Yan Duan & Ziming Li
Brian Prike & Zhenghao Qian
Parker Schuh & Robert Chatham

Fall 2012 Winners
Chenyang Yuan
Joseph Hui

Fall 2013 Winners
Paul Bravero
Sam Kumar & Kangkuk Lee
Kevin Chen

Fall 2014 Winners
Alan Tong & Elaine Zhao
Zheyuan Zhang
Adam Robert Villafior & Joany Gao
Zhiyong Li & Tian Li

Spring 2015 Winners
Sinho Chewi & Alexander Nguyen Tran
Zhaowei Li
Stella Tao and Yao Ge

Fall 2015 Winners...

Order of Recursive Calls

The Cascade Function

```python
def cascade(n):
    if n < 10:
        print(n)
    else:
        print(n)
        cascade(n // 10)
        print(n)

cascade(123)
```

Program output:
```
1
2
3
4
12
34
123
```

Two Definitions of Cascade

```python
def cascade(n):
    if n < 10:
        print(n)
    else:
        print(n)
        cascade(n // 10)
        print(n)
```

Order of Recursive Calls

“- If two implementations are equally clear, then shorter is usually better
- In this case, the longer implementation is more clear (at least to me)
- When learning to write recursive functions, put the base cases first
- Both are recursive functions, even though only the first has typical structure

Inverse Cascade

Write a function that prints an inverse cascade:

```python
def inverse_cascade(n):
    if n:
        def f_then_g(f, g, n):
            if n:
                g(n)
                f(n)
            else:
                g(n)

        grow = lambda n: f_then_g(lambda n: print(n), n // 10)
        shrink = lambda n: f_then_g(lambda n: print(n), n // 10)

        grow(n)
        shrink(n)
```

Example: Inverse Cascade

```
1
2
3
4
12
34
123
```
Tree Recursion

Tree-shaped processes arise whenever executing the body of a recursive function makes more than one recursive call.

\[ a: \quad 0, 1, 2, 3, 4, 5, 6, 7, 8, \ldots , 15 \]
\[ \text{fib}(n): \quad 0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots , 9,227,465 \]

```python
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-2) + fib(n-1)
```

A Tree-Recursive Process

The computational process of fib evolves into a tree structure.

```plaintext
fib(5)
   /   
/     
fib(3)  fib(4)
     /   
/     
fib(2)  fib(2)
     /   
/     
fib(1)  fib(1)
     /   
/     
0  1
```

Repetition in Tree-Recursive Computation

This process is highly repetitive; fib is called on the same argument multiple times.

```plaintext
fib(5)
   /   
/     
fib(3)  fib(2)
     /   
/     
fib(1)  fib(1)
     /   
/     
0  1
```

Counting Partitions

The number of partitions of a positive integer \( n \), using parts up to size \( m \), is the number of ways in which \( n \) can be expressed as the sum of positive integer parts up to \( m \) in increasing order.

```plaintext
\[
\begin{align*}
2 + 4 &= 6 \\
1 + 1 + 4 &= 6 \\
3 + 3 &= 6 \\
1 + 2 + 3 &= 6 \\
1 + 1 + 1 + 3 &= 6 \\
2 + 2 + 2 &= 6 \\
1 + 1 + 2 + 2 &= 6 \\
1 + 1 + 1 + 1 + 2 &= 6 \\
1 + 1 + 1 + 1 + 1 + 1 &= 6 \\
\end{align*}
\]
```

Example: Counting Partitions

```python
def count_partitions(n, m):
    if n == 0:
        return 1
    elif n < 0:
        return 0
    elif m == 0:
        return 0
    else:
        with_m = count_partitions(n-m, m)
        without_m = count_partitions(n, m-1)
        return with_m + without_m
```

Counting Partitions

The number of partitions of a positive integer \( n \), using parts up to size \( m \), is the number of ways in which \( n \) can be expressed as the sum of positive integer parts up to \( m \) in increasing order.

```plaintext
\[
\begin{align*}
\text{count_partitions}(6, 4)
\end{align*}
\]

Recursive decomposition: finding simpler instances of the problem.
- Use at least one 4
- Don't use any 4
- Solve two simpler problems:
  - count_partitions(2, 4) 
  - count_partitions(6, 3)
- Tree recursion often involves exploring different choices.