CS 61A Discussion 2
Environment Diagrams / Recursion

ANNOUNCEMENTS!!!

Extra environment diagram questions on the worksheet linked from the course website (cs61a.org)!

Hopefully you’ve finished (or are almost finished with) Hog by now… because it’s due tonight!

Online office hours are now a thing!

HW2 due next week! Party in 247 Cory next Monday (6:30-8:30pm)!

MORE ANNOUNCEMENTS.

CSM signups on Piazza. Expanding sections will have signups available starting today at 1pm.

Register for our CS 198 class if you're interested in furthering your knowledge about general computer-scientific concepts.

Do not post code publicly on Piazza. Or anywhere, really.

ATTENDANCE

```
if disc == 116:
    use(‘curvatureflow’)
else:
    use(‘simplicialcomplex’)
```

1.

Quiz II / Env. Diagrams II

Let’s see how much you remember about environment diagrams…!

Challenge Questions

for those who want a challenge

CHALLENGE QUESTION #1

Fill in the blanks (without using any numbers in the first blank!) to get the expected output.

```
>>> (lambda x: lambda y: _______)(____)(lambda z: z * z)()
9
```

More environment diagrams

<see extra set of slides>

2.

Recursion

Defining functions in terms of themselves

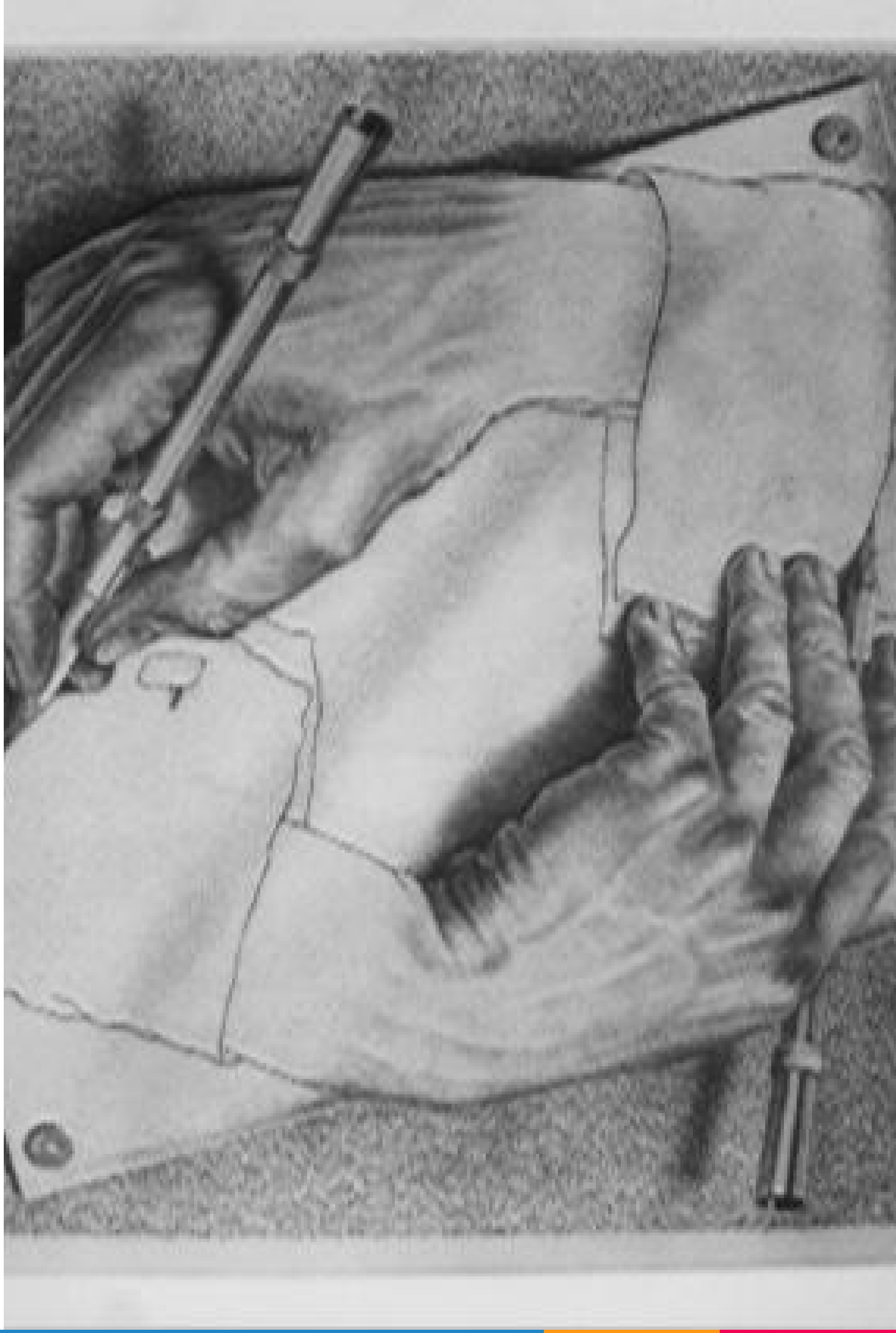
“To understand recursion, one must first understand recursion” - Darth Vader

RECURSION

Recursion is where you call a function from its own body.

```
def stack(overflow):
    return stack(overflow)
```

It's technically recursion… but what is wrong with the function above?



Your base case

is the smallest or simplest case.

Your recursive case

defines the problem in terms of a simpler problem.

Tie the two together

by working toward the base case in your recursive calls.

Why does recursion work? Let's prove the correctness of one example using induction…

show that the base case works → assume that for an arbitrary valid input, the function works → prove that for an input of one unit greater size, the function also works (i.e. you've properly tied the recursive calls together)

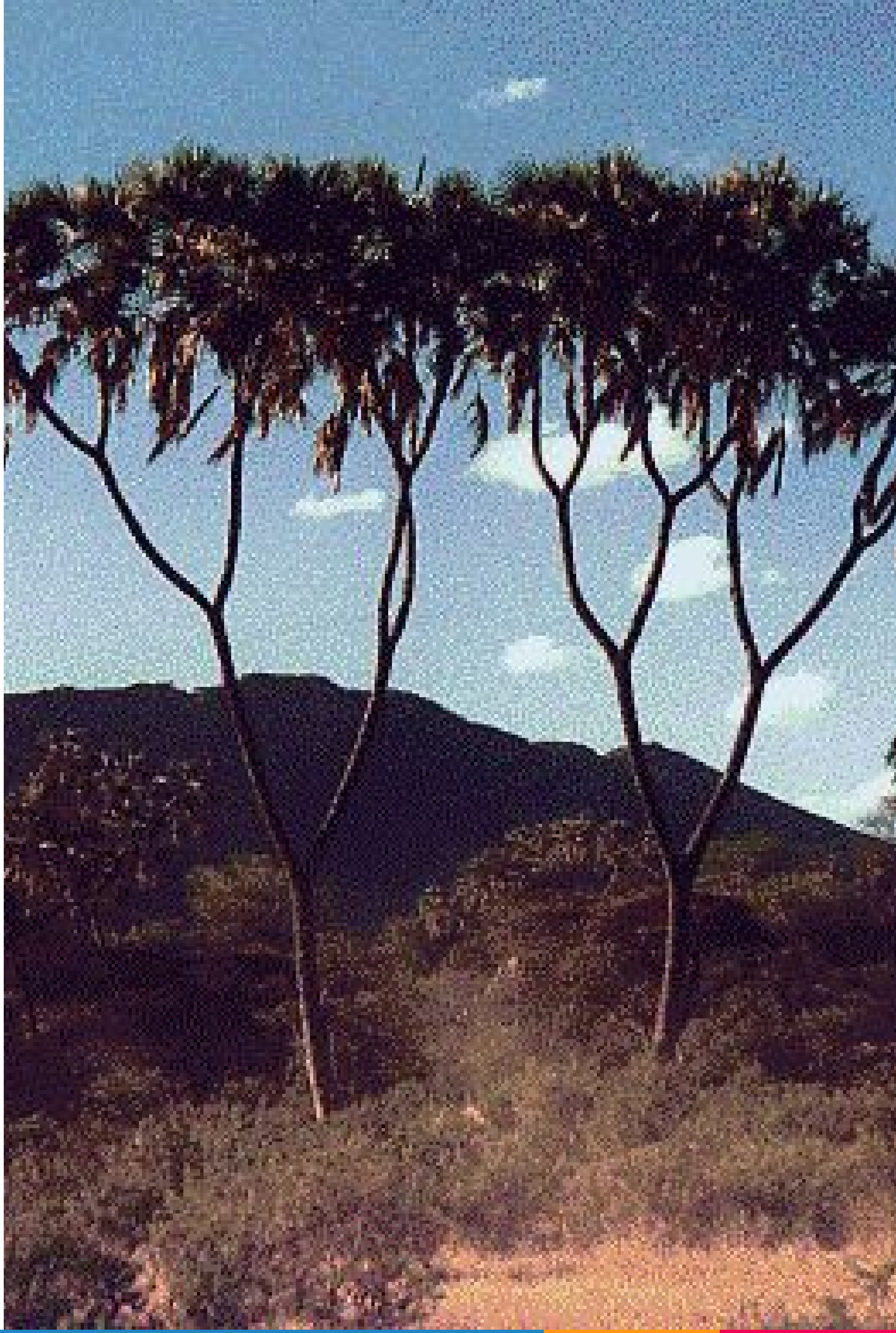
If we can prove the above, we've proved correctness for all inputs. Thus, in practice we can take the "recursive leap of faith" under security of these types of proofs.

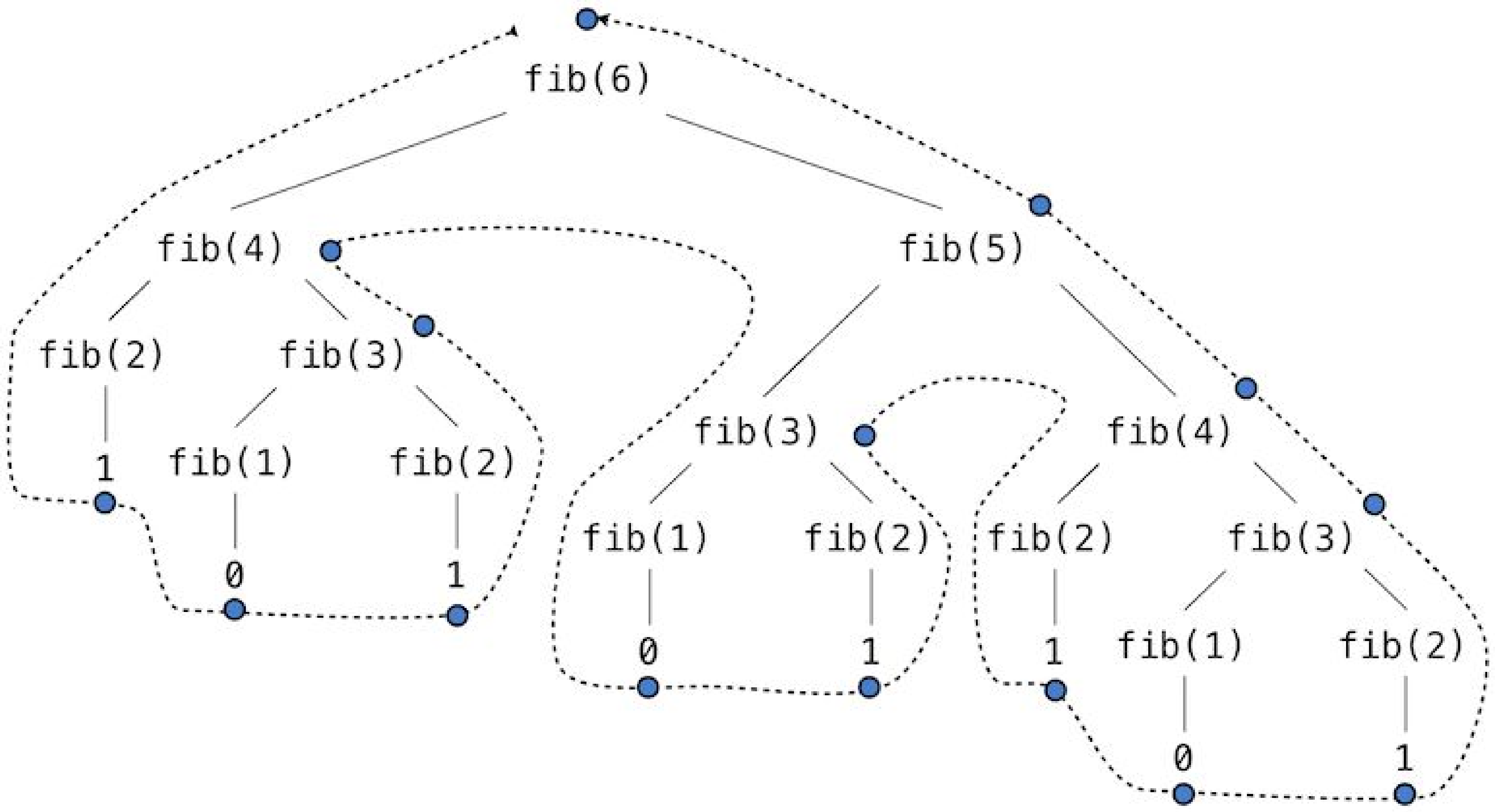
TREE RECURSION

Tree recursion is where you make *more than one recursive call* in a single function call.

```
def fib(n):
    if n <= 1:
        return n
    return fib(n - 1) + \
          fib(n - 2)
```

(See how there are two recursive calls in the body above!)





The Fibonacci recursion tree. The blue dots depict the order in which the function calls return.

また来週！

Thanks, everyone...