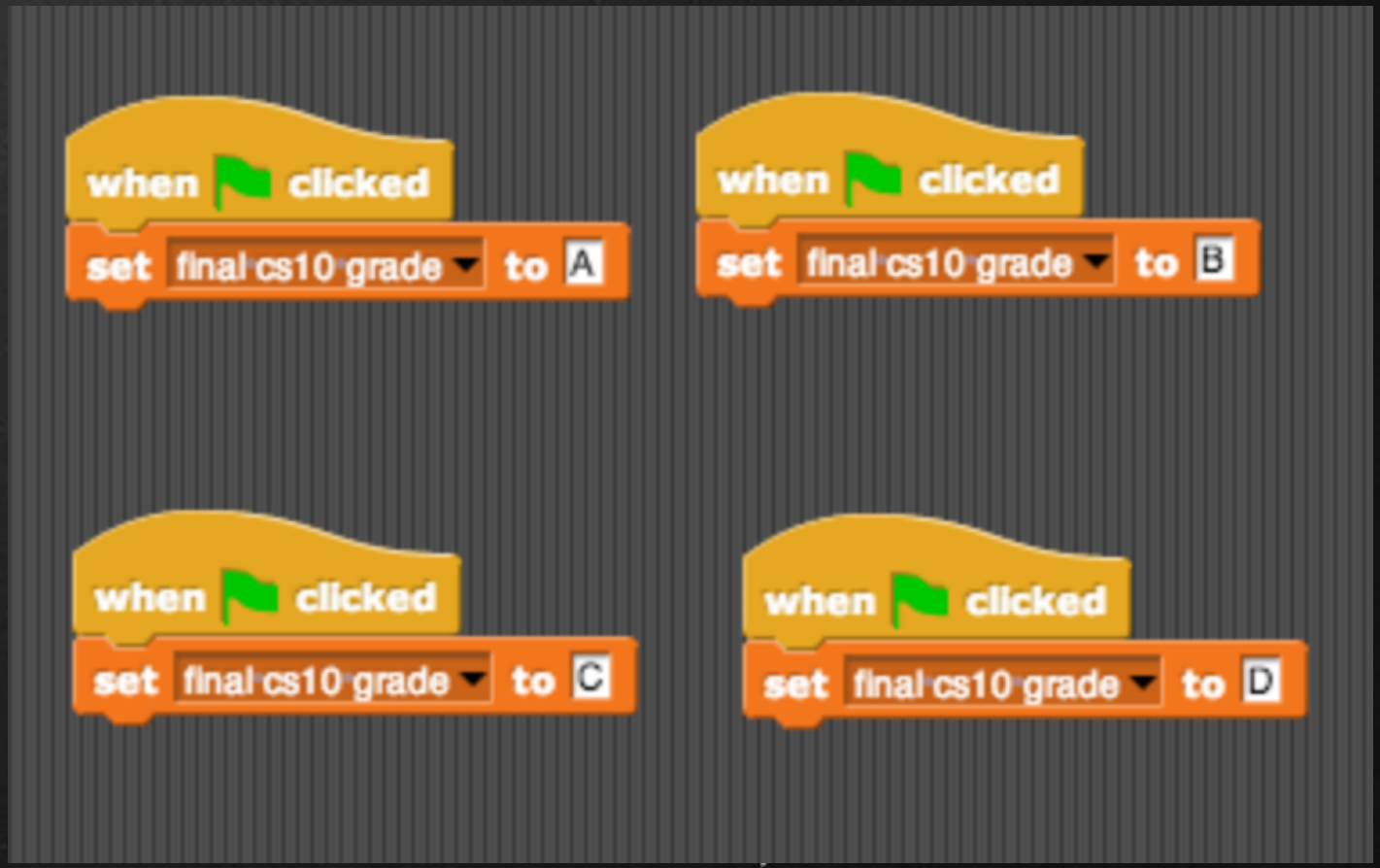
Concurrency

Concurrency

Definition: Several scripts are executing simultaneously and potentially interacting with each other



This is how we assign grades! Based on the Birkahni Theorem, we usually get the grades to average to a B+, though due to the size of the class this semester, the average will be a C+ (jk)

Race Condition

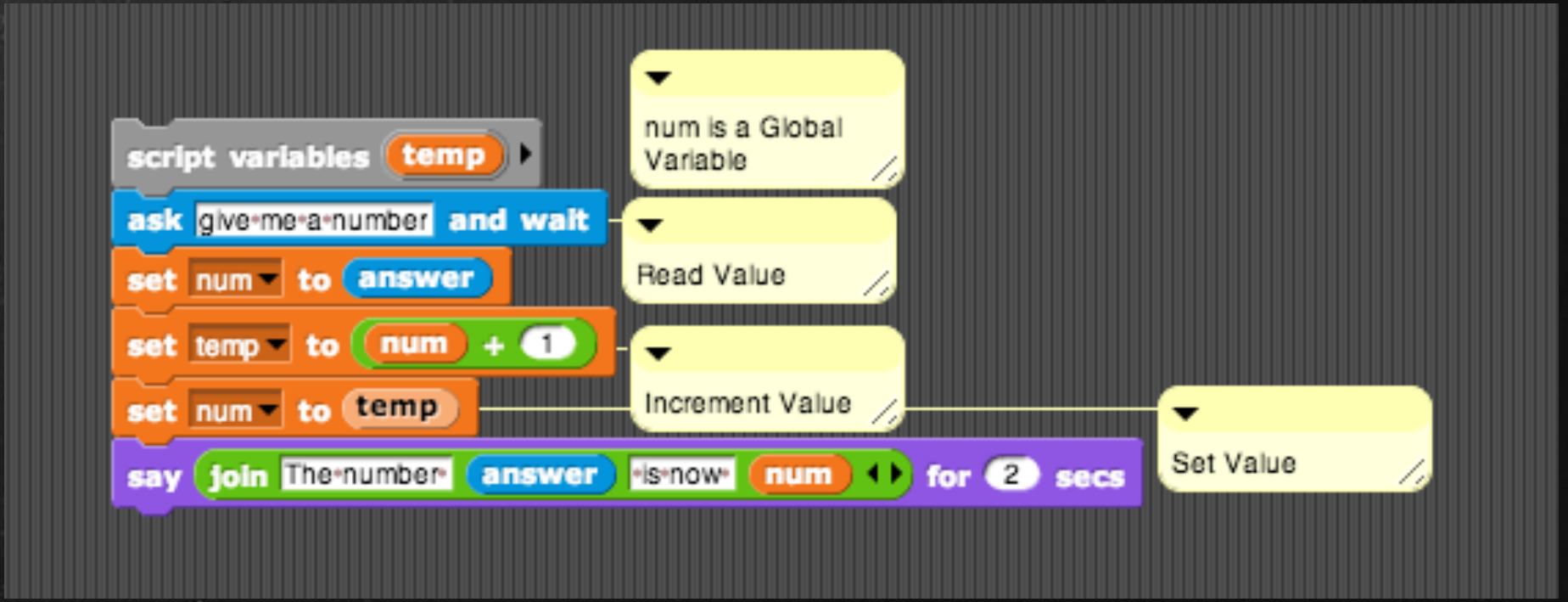
Concurrency Issue

Race Condition

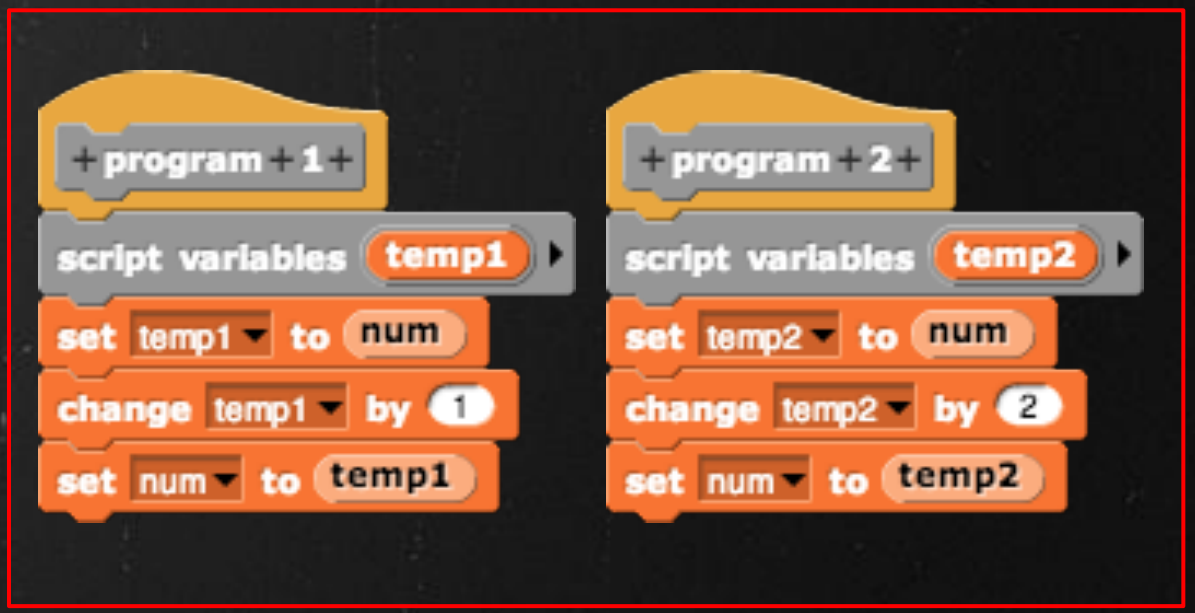
Definition: when events of a program don't happen in the order that the programmer intended.

Function Definitions

- read value: reads in a value from user input
- increments value: increments the value, but does not set it
- sets value: sets the value to the incremented version of it.



Race Condition Example



We have two programs, program 1 and program 2, and a global variable ‘num’. Ideally, we want the script in program 1 to run before the script in program 2, but this won’t always be the case. We’ll look at two scenarios, the first where they run in order (serial), and the second where they don’t (race condition).

Serial - Example

program 1	program 2	num (Global Integer Value)
		0

‘num’ starts out with value 0

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0

read value

(reads the value of ‘num’, and sets ‘temp1’ to that value)

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
change temp1 by 1		0

<u>increments value</u>
(increases the value of ‘temp1’ by 1)

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
change temp1 by 1		0
set num to temp1		1

<u>sets value</u>

(sets ‘num’ to the value of ‘temp1’, which is 1)

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
change temp1 by 1		0
set num to temp1		1
	set temp2 to num	1

read value

(reads the value of ‘num’, and sets ‘temp2’ to that value)

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
change temp1 by 1		0
set num to temp1		1
	set temp2 to num	1
	change temp2 by 2	1

<u>increments value</u>
(increases the value of ‘temp2’ by 2)

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
change temp1 by 1		0
set num to temp1		1
	set temp2 to num	1
	change temp2 by 2	1
	set num to temp2	3

<u>sets value</u>

(sets ‘num’ to the value of ‘temp2’, which is 3)

Serial - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
change temp1 by 1		0
set num to temp1		1
	set temp2 to num	1
	change temp2 by 2	1
	set num to temp2	3

This is the expected output. We're good here!

What if we interleaved the commands?

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0

‘num’ starts out with value 0

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0

read value

(reads the value of ‘num’, and sets ‘temp1’ to that value)

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
	set temp2 to num	0

read value

(reads the value of ‘num’, and sets ‘temp2’ to that value)

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
	set temp2 to num	0
change temp1 by 1		0

<u>increments value</u>
(increases the value of ‘temp1’ by 1)

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
	set temp2 to num	0
change temp1 by 1		0
	change temp2 by 2	0

<u>increments value</u>
(increases the value of ‘temp2’ by 2)

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
	set temp2 to num	0
change temp1 by 1		0
	change temp2 by 2	0
set num to temp1		1

<u>sets value</u>

(sets ‘num’ to the value of ‘temp1’, which is 1)

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
	set temp2 to num	0
change temp1 by 1		0
	change temp2 by 2	0
set num to temp1		1
	set num to temp2	2

<u>sets value</u>
(sets ‘num’ to the value of ‘temp2’, which is 2)

Race Condition - Example

program 1	program 2	num (Global Integer Value)
		0
set temp1 to num		0
	set temp2 to num	0
change temp1 by 1		0
	change temp2 by 2	0
set num to temp1		1
	set num to temp2	2

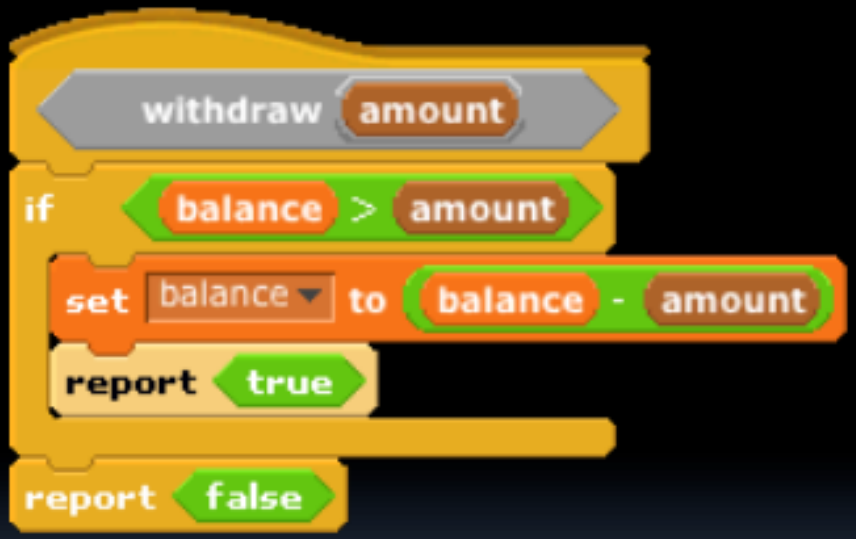
This is the NOT the expected output. ‘num’ is only 2!

Takeaway

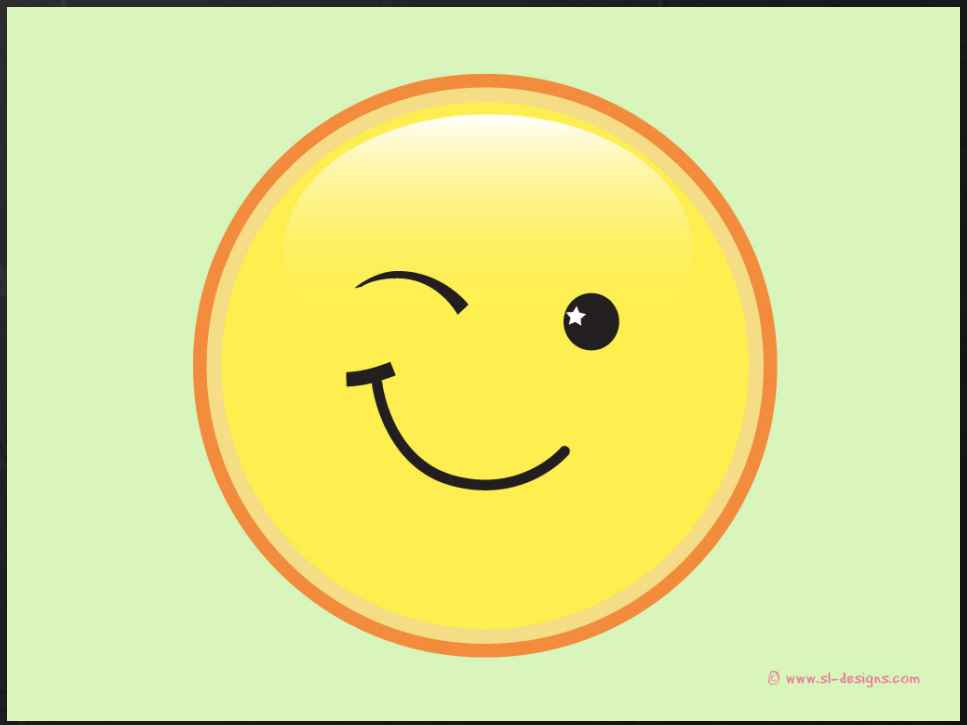
Concurrency is great because it allows for tasks to be broken up and completed almost simultaneously. However, you have to be careful how you break up the tasks so you don’t get erroneous behavior.

Race Condition Example from Lecture

- **What if two people were calling withdraw at the same time?**
  - E.g., balance=100 and two withdraw 75 each
  - Can anyone see what the problem *could* be?
  - This is a race condition
- **In most languages, this is a problem.**
  - In Scratch, the system doesn't let two of these run at once.



Winky Face Problem



Question 13: Your faaaaace... (5 pts)

You want to draw a face, so you write this serial script that produces the "winking" face right beside it:

But then you want to simulate what it would be like to parallelize the code and run it on three separate "cores", so you change the serial script above into the following parallel scripts, which all run at the same time:

Draw all the faces that could result from running this new parallel code. You may not need all the blanks.

Question 13: Your faaaaace... (5 pts)

You want to draw a face, so you write this serial script that produces the "winking" face right beside it:

But then you want to simulate what it would be like to parallelize the code and run it on three separate "cores", so you change the serial script above into the following parallel scripts, which all run at the same time:

Question 12/13: *Draw all the faces* that could result from running this new parallel code. You may not need all blanks. These result from interlacing 3 LeftEye/RightEye/Mouth Clear (LC,RC,MC), LeftEye(L), RightEye(R), & Mouth(M).

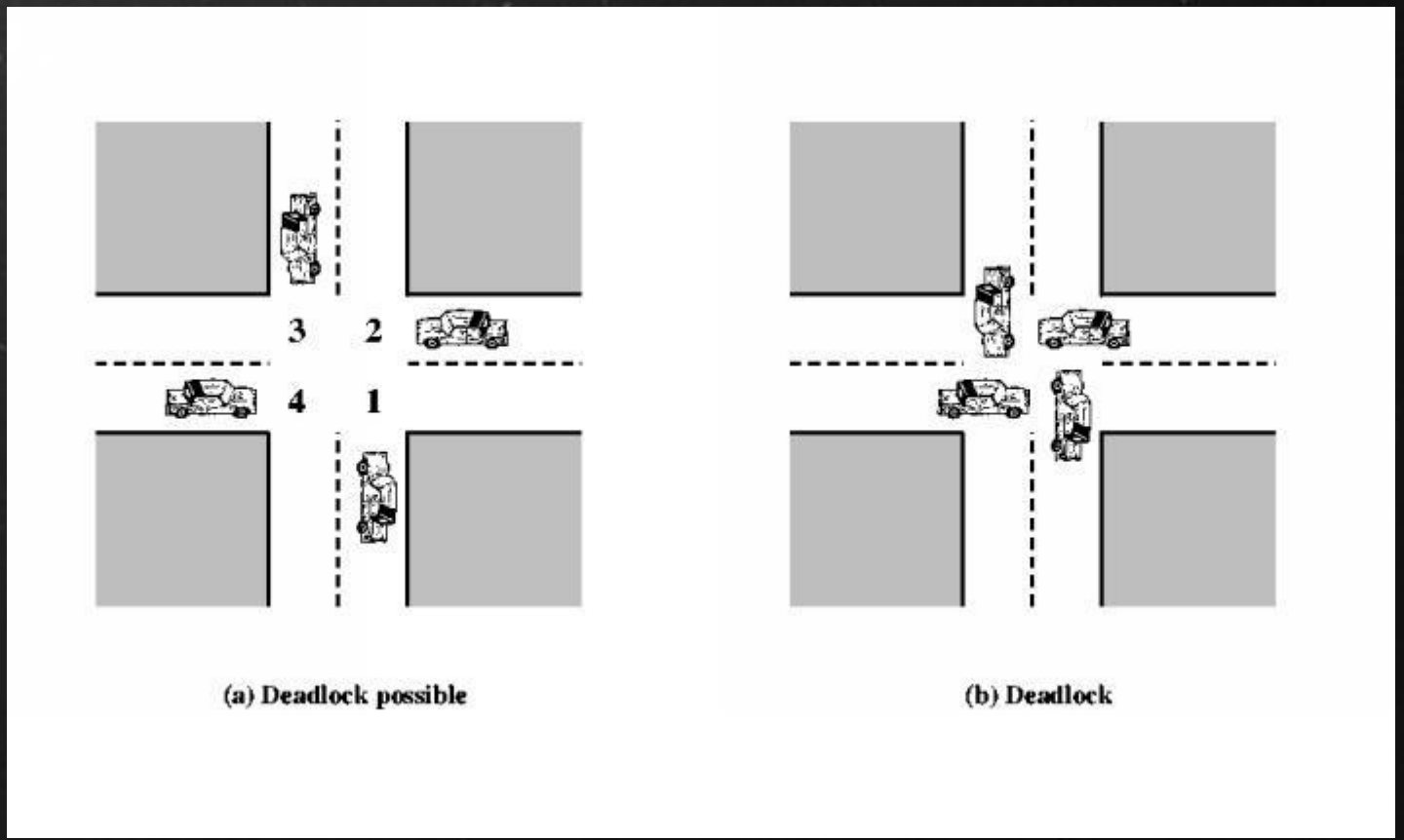
 RC,R,MC,M,LC,L	LC,L,MC,M,RC,R	 LC,L,RC,R,MC,M	 RC,R,MC,LC,M,L	 LC,L,MC,RC,M,R	 MC,M,LC,RC,L,R	 RC,LC,MC,R,L,M

Deadlock

Concurrency Issue

Deadlock

Definition: a situation in which two or more competing actions are each waiting for the other(s) to finish, and thus no one ever finishes.



Deadlock - Example

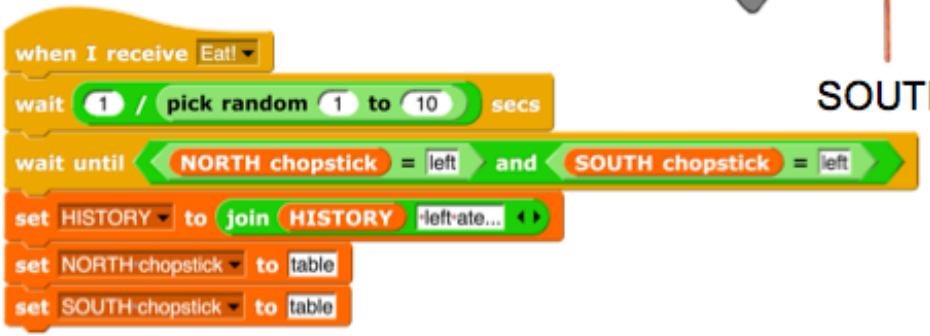
Dining Philosopher Problem

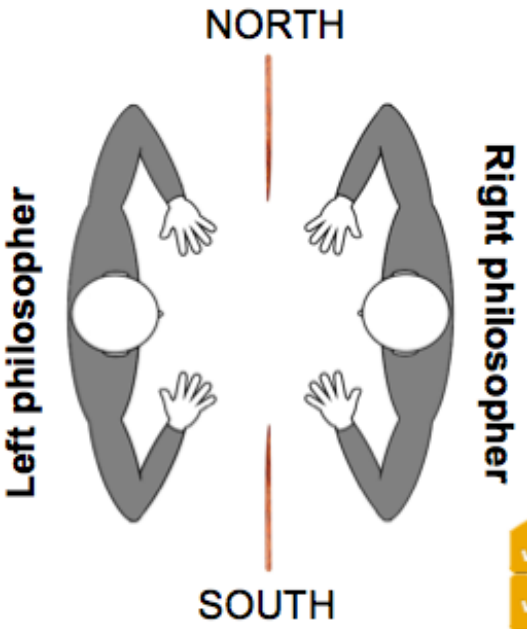
SID: _____

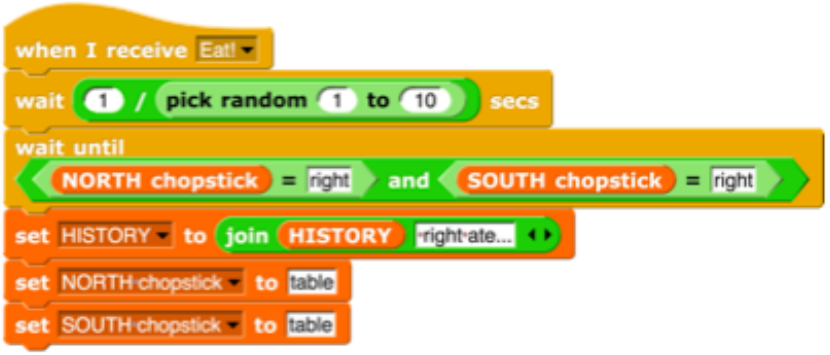
Question 12: *Dining Philosophers* (5 pts)

Two philosophers (left and right) are having dinner, sitting across from each other. There is a NORTH and a SOUTH chopstick on the table. Each philosopher continually looks down to see if a chopstick is on the table, and tries to grab it; if both are ever grabbed by one person, that person eats, updates HISTORY (a record of what happened) and puts the chopsticks down. Ten seconds after the green flag is clicked, what could HISTORY be?

(all the boxes are not necessarily needed)







SID: _____

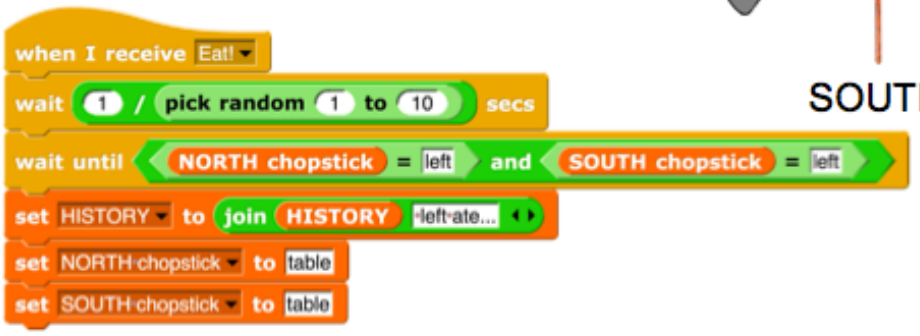
Share

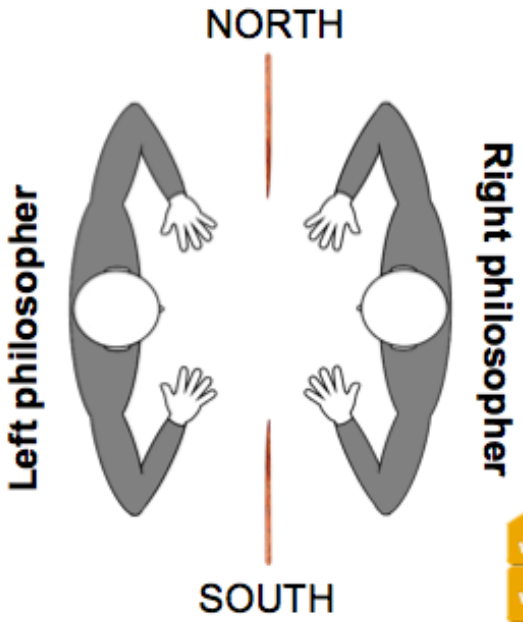
Question 12: *Dining Philosophers* (5 pts)

Two philosophers (left and right) are having dinner, sitting across from each other. There is a NORTH and a SOUTH chopstick on the table. Each philosopher continually looks down to see if a chopstick is on the table, and tries to grab it; if both are ever grabbed by one person, that person eats, updates HISTORY (a record of what happened) and puts the chopsticks down. Ten seconds after the green flag is clicked, what could HISTORY be?

(all the boxes are not necessarily needed)

Started… **Left ate…** **Right ate…**	**Started…** **Right ate…** **Left ate…**	**Started…**		





Recursion

Base Case(s):

Recursive Case(s):

Recursion

Base Case(s):
-Simplest form of the problem

Recursive Case(s):

Recursion

Base Case(s):
-Simplest form of the problem

Recursive Case(s):
-Divide problem into smaller instances

Recursion

Base Case(s):
-Simplest form of the problem

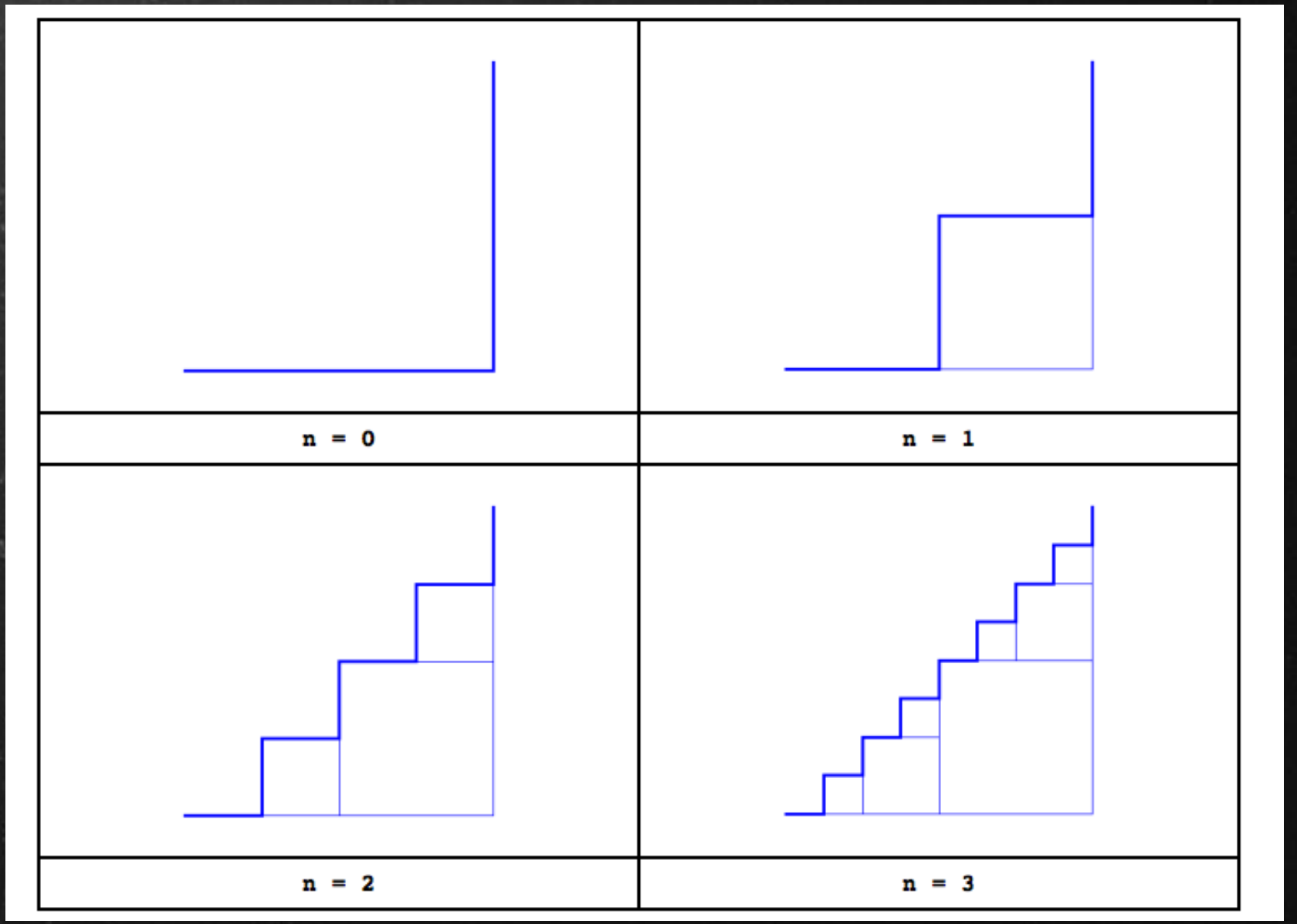
Recursive Case(s):
-Divide problem into smaller instances
-Invoke function (recursively)

Recursion

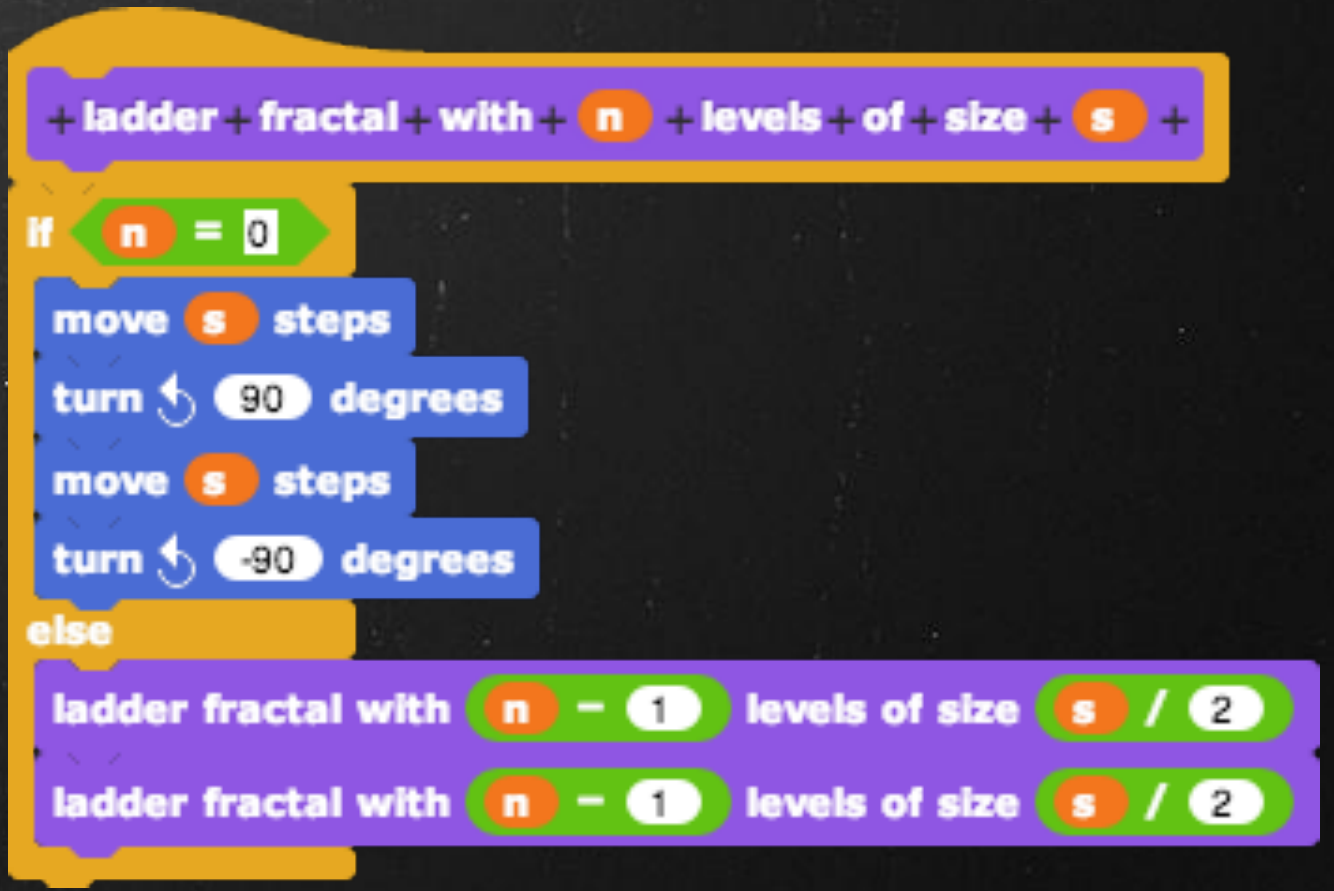
Base Case(s):
-Simplest form of the problem

Recursive Case(s):
-Divide problem into smaller instances
-Invoke function (recursively)
-Work towards base case

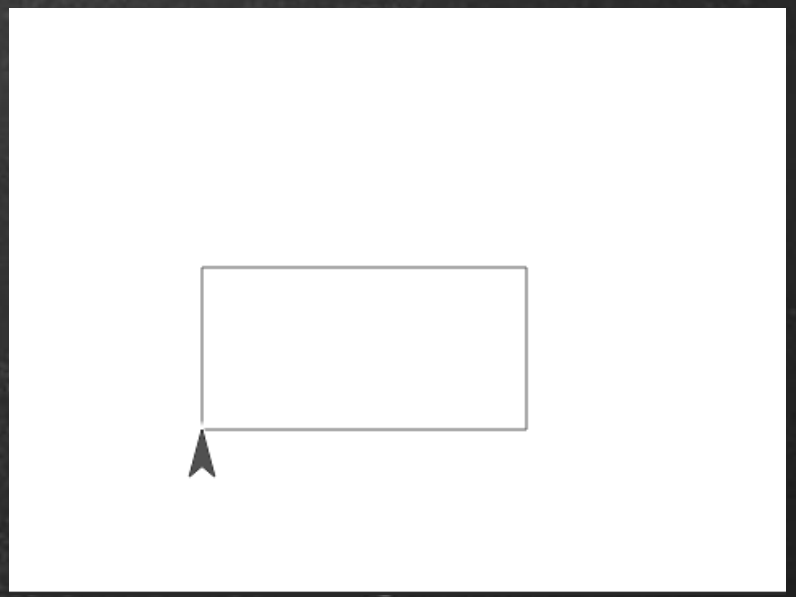
Ladder Fractal

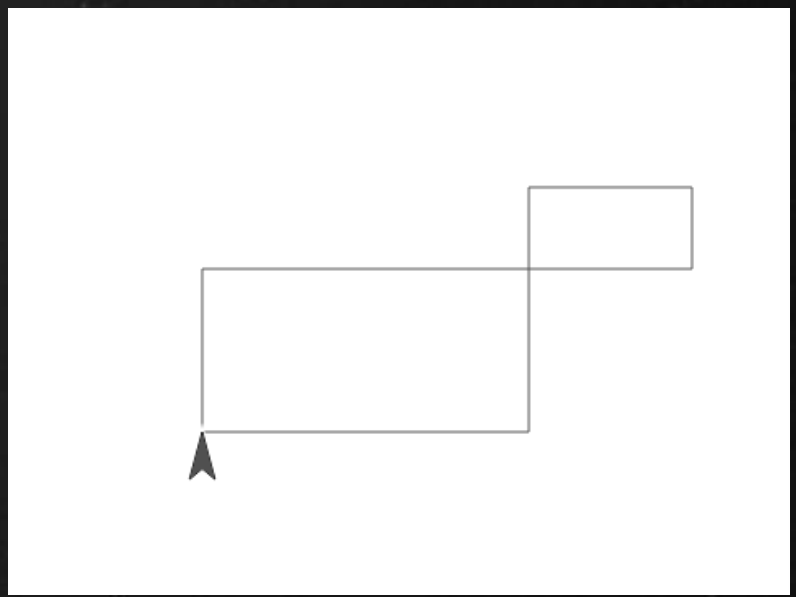


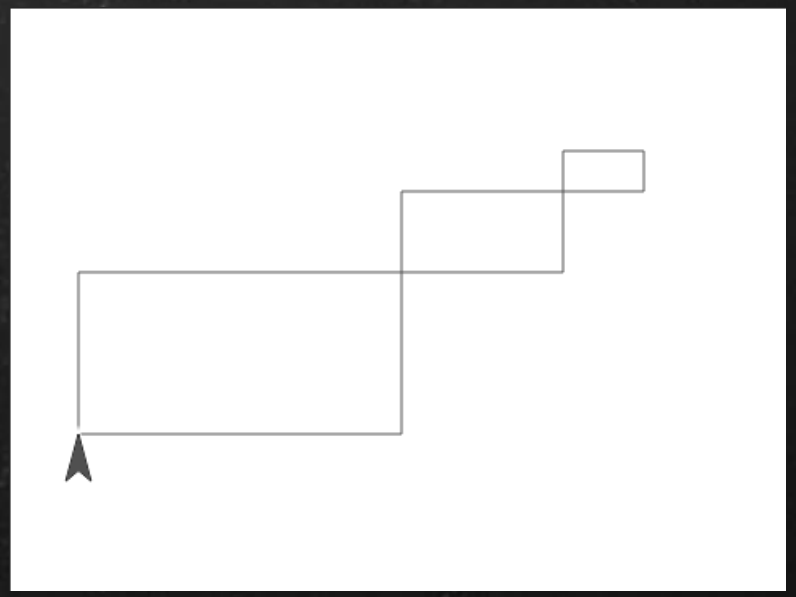
Ladder Fractal Solution

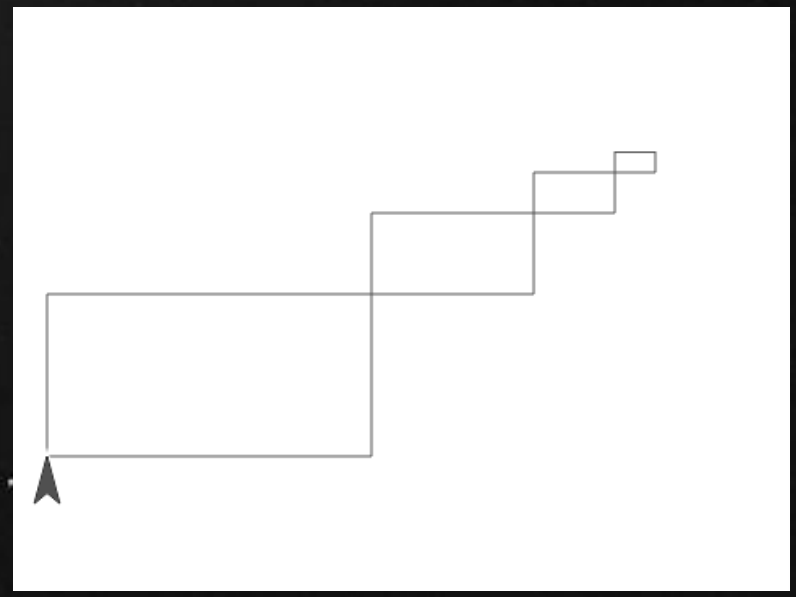


Rectangle Fractal









Rectangle Fractal Solution