



Quest Review Session

CS10 Fall 2015

-Andy, Carlos, Joseph



Follow us online!

The slides: <http://bit.ly/1LHT5ok>



Stuff to Know

The quest is Wednesday, September 30, 2015. It is in class.

Get there early so we can begin right away.

You get one 8.5 x 11 **hand-written** cheat sheet. You can write on the front and back.

You do not need to memorize blocks. You should know the common blocks. (Ex. if we ask you to use a Higher Order Function to solve a problem, you should know that you'll be using map, keep or combine.)



What's on it?

Anything from lectures, labs, readings, and discussions (except computer hardware) is fair game.

Some topics:

- Abstraction
- Functions
- Programming paradigms
- Turing
- Algorithms
- Lists
- HOFs (Map, Keep, and Combine)



The Plan

Things we'll cover today:

Readings

Other Lecture Topics

Programming Concepts

Practice Problems



Readings



Brian Harvey's Introduction to Abstraction

Main point: abstraction is intentionally hiding information that is unnecessary for others to know behind a more convenient *interface*.

Programming: create functions that do something but don't necessarily reveal *how* it's done.

Why is this useful?

- Makes advanced ideas and tools accessible to those who don't fully understand them
- Designers are able to change how things work internally without changing the interface.
- Generalization (same interface across multiple tools).



Why Software is Eating the World

Main point: software-based businesses are dominating in many sectors of the economy.

Software isn't only for the Googles and Microsofts of the world. It's showing up in almost every sector of the economy, changing the way that business operates.

Software isn't falling off the radar any time soon.

Ex) Amazon eating Borders, Netflix eating Blockbuster, etc



Learning to Code!

Main point: having basic programming skills is the equivalent to literacy in the 21st century.

The world is increasingly integrated with computer technology, and knowing nothing about how it works is somewhat dangerous.

Using a word processor or a web browser is using technology in a very limited way. It's like only being able to read what others write and not being able to write yourself.

Watch the ted talk again, it's 13 minutes.



The First Object Oriented Software System

- Ivan Sutherland makes SketchPad in 62-63 for his thesis
- Object Oriented programming is now ingrained in most major languages. (c++, java, python, javascript...)
- OOP allows you to create objects, and have many of them, each instance having the same characteristics! (like cloning in Snap!)
- The light pen was a bad invention



The Story of Alan Turing and his Machine

- Considered by many to be “Father of Computer Science”
- Helped crack german enigma machine during WWII
- Turing’s conceptual computational machine
 - endless tape (lol, tape) can only read/write, move left or right
 - each place on the tape contains only 1 symbol
 - moves and edits tape as the program instructs
- at any given time computers can only have 1 state, so this idea, while old, still applies
- Universality - computers can be made with the same machinery but be able to execute any program!!!!!!!!!!!!!!
- (^ this is huge)



Algorithms

Shape our world: Algorithms are everywhere, and sometimes we don't know what they do or we have lost track of old ones. (brief rant on stuxnet/ stock market)

Can ruin lives: Can have unintended consequences. Dropping people from insurance, racial profiling in facial recognition software. (Humans make these. Solution?)

Can ruin lives comments: Many tend to agree that some algorithms cause problems and giving them free reign without oversight might not be the way to go, but there is much debate over how to combat this issue.



How Algorithms Shape Our World

Big idea: algorithms are everywhere, and they're active, not passive

Examples-

Finance: Wall Street; black box/algorithmic trading

Physical: Chicago-NY fiber-optic cable

Culture: Netflix's "pragmatic chaos"; Epagogix

More: elevators, Roomba, etc.

The problem:

We're "writing the unreadable."

Algorithms locked together without "human oversight"

\$23 million book on Amazon...?



Blown to Bits, Chapter 1

Main point: technology is neither good nor bad.

Opportunities Vs. Risks

Nuclear reactions & energy:

Pros: creates electric power

Cons: creates weapons of mass destruction

Moore's Law

Number of transistors that can fit into a specific area will double every 18 - 24 months.

The Seven Koans -- what are they and why are they important?



Blown to Bits, Chapter 1

7 Koans:

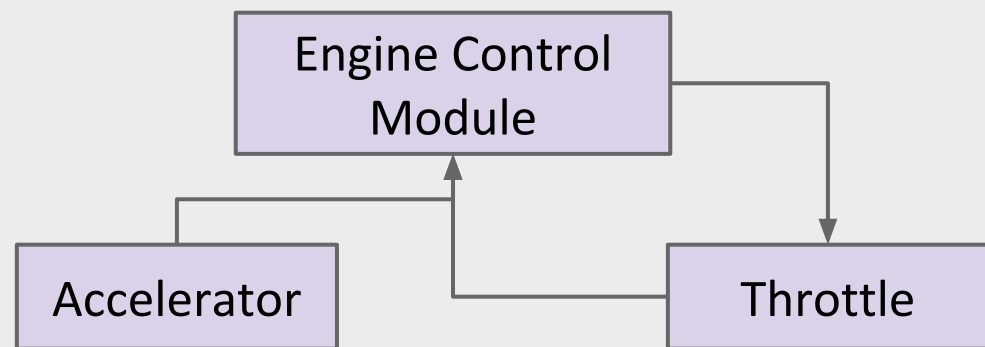
- 1. It's All Just Bits**
- 2. Perfection is Normal**
- 3. There is want in the midst of plenty**
- 4. Processing is Power**
- 5. More of the same can be a whole new thing**
- 6. Nothing Goes Away**
- 7. Bits move faster than thought**



Other Lecture Topics

Lecture 1: Abstraction

- Generalization (functions, bits)
- Detail Removal (train maps, cliff notes)
- Modularization (separation of concerns)
- Make sure you can think of your own examples of abstraction!!!





Lecture 2: Procedures

Procedure: A sequence of instructions that are packaged into a single unit

- Generalization of Code
- Allow us to break a problem down into smaller parts that are more manageable and testable (a.k.a. functional decomposition)

Lecture 2: Procedures

Functions: A special type of procedure that has no side effects

- check out slide 16 for all the specifics
- Why is the procedure 'is mario big?' not a function?





Lecture 3: Numbers/Abstraction

Main takeaways:

- Computers can most easily understand something being on or off, which is why we use bits to represent things (1 - on, 0 - off)
- We'll do some conversion examples later



Lecture 4: Programming Paradigms

A formal definition:

- The concepts and abstractions used to represent the elements of a program (e.g., objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignment, evaluation, continuations, data flows, etc.).

The main idea:

- “A programming paradigm is a general approach, orientation, or philosophy of programming that can be used when implementing a program.”



Lecture 4: Programming Paradigms

1) Functional

- evaluate an expression and use the result
- remember that functions do not have side effects!

2) Imperative

- follow a list of instructions one by one
- Think of a recipe, or a simple script that moves the sprite around in snap

3) Object-Oriented (example: sketchpad)

- construct instances from classes and send messages between them

4) Declarative (ex: a physics simulator, coloring a map)

- answer a question via search for a solution

ALL ARE EQUALLY POWERFUL



Lecture 5: Algorithms

An algorithm is a tool for solving a computational problem:

- An algorithm describes a specific computational procedure to get a desired result from a certain input
- A function may use an algorithm, but an algorithm is not a function
- A function takes in inputs and gives outputs, but an algorithm describes the process that happens inside a function



Lecture 9: Cryptography

Why do we need it?

- Almost all channels of communication are insecure so we need to find a way to send messages securely
- Encryption solves this problem by obscuring the original message as it travels. If we are the person who was supposed to receive the message then we know how to decipher it
- Review specific types of encryption from the slides (substitution ciphers, RSA, etc...)
- Just try to get the main ideas



Programming Concepts



Variables

Variables are used to store values that we need the computer to remember for later.

When we use a variable in a script, the computer looks up the value of the variable and substitutes it into the script. This look-up is performed every time we use a variable, and is called **evaluation**.

What's a script variable? When would we use one?



Conditionals (if / else)

Conditionals are the way to make a program react to different situations. They exist in *Snap!* as the "if" and "if-else" blocks.

The "if" block will only execute the code within its C-shape if the predicate (hexagonal block) that it has been given evaluates to "true". This predicate is called the **condition**, and has to be a boolean expression.

The "if-else" block will execute either one or the other pieces of code that you give to its two C-shapes. This should only be used when a situation needs to be handled a completely different way for either case.



Functions: yes or no

Which of the following blocks are functions?





Lists

What is an index?






What happens to a list if you add an item to it?
(how does its length and index change?)

What happens if you delete an item in the middle of a list?
(does the index of some items change? if so, which ones?)

What is the length of a list if the list is empty?



Higher Order Functions

INPUT	DESIRED OUTPUT	MAP KEEP COMBINE	FUNCTION
List of length n	List of length n	Map	 
List of length n	List of length $\leq n$	Keep	
List of length n	Same type as the items inside the input list	Combine	 

Higher Order Functions



Higher Order Functions



So can keep.



Higher Order Functions



Combine MUST have exactly 2 blank inputs.

Higher Order Functions

In general, HOFs can take in both reporters and predicates as their function input.



You can test these out for yourself. What do you notice?

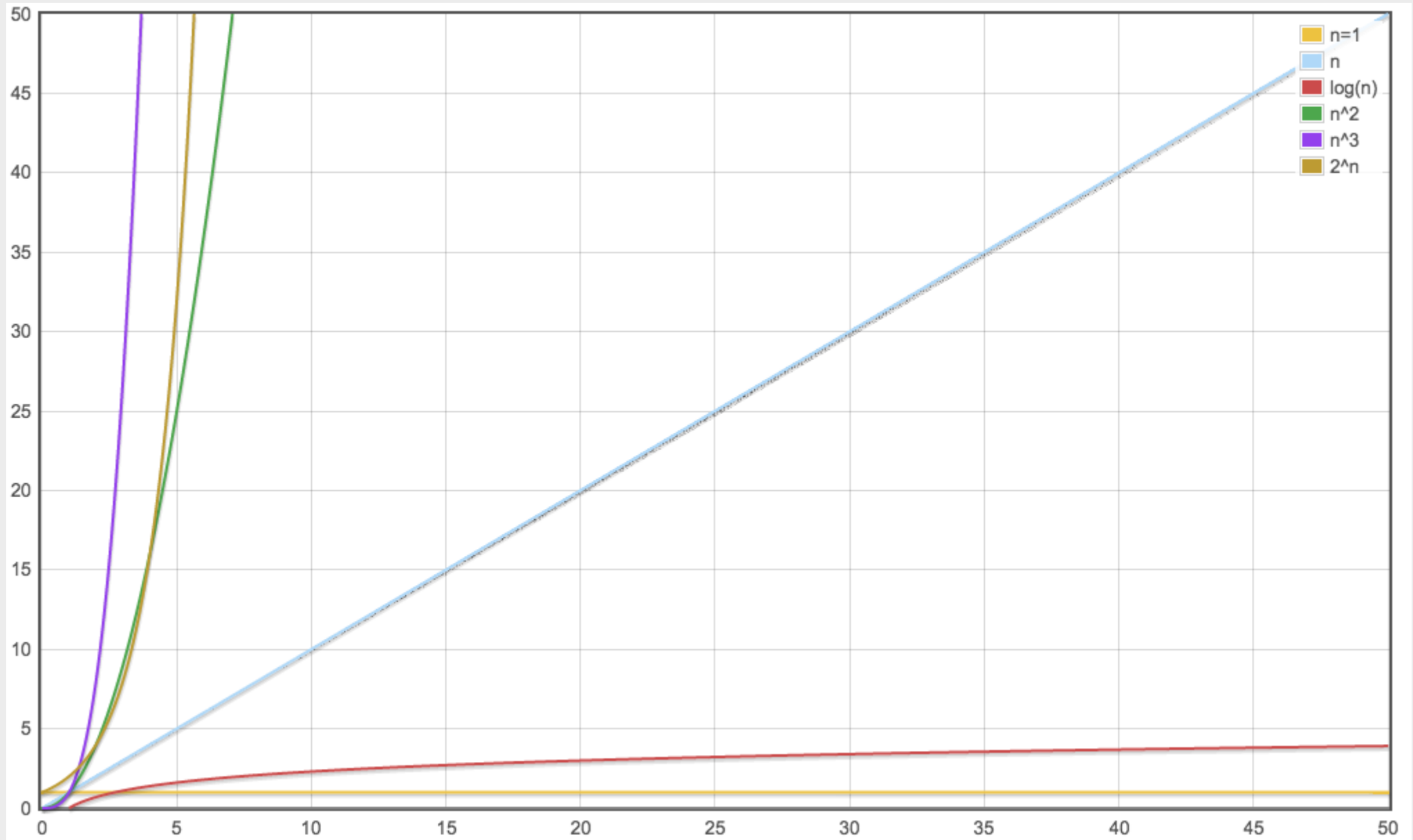
Algorithmic Complexity

"Algorithmic Complexity", also called "Running Time" or "Order of Growth", refers to the number of steps a program takes as a function of the size of its inputs. We will say this input has length n .





Algorithmic Complexity





Algorithmic Complexity

Runtime	Big-O notation
Constant	$O(1)$
Logarithmic	$O(\log n)$
Linear	$O(n)$
Quadratic	$O(n^2)$
Cubic	$O(n^3)$
Exponential	$O(2^n)$



Practice Problems



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal.

Convert 101_{10} from decimal to binary.

Convert 1011100_2 from binary to hex.

Convert AC_{16} from hex to binary.

Convert 43_{10} from decimal to hex.

Convert 43_{16} from hex to decimal.

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary.

Convert 1011100_2 from binary to hex.

Convert AC_{16} from hex to binary.

Convert 43_{10} from decimal to hex.

Convert 43_{16} from hex to decimal.

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex.

Convert AC_{16} from hex to binary.

Convert 43_{10} from decimal to hex.

Convert 43_{16} from hex to decimal.

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex. $\rightarrow 5C_{16}$

Convert AC_{16} from hex to binary.

Convert 43_{10} from decimal to hex.

Convert 43_{16} from hex to decimal.

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex. $\rightarrow 5C_{16}$

Convert AC_{16} from hex to binary. $\rightarrow 10101100_2$

Convert 43_{10} from decimal to hex.

Convert 43_{16} from hex to decimal.

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex. $\rightarrow 5C_{16}$

Convert AC_{16} from hex to binary. $\rightarrow 10101100_2$

Convert 43_{10} from decimal to hex. $\rightarrow 101011_2 \rightarrow 2B_{16}$

Convert 43_{16} from hex to decimal.

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex. $\rightarrow 5C_{16}$

Convert AC_{16} from hex to binary. $\rightarrow 10101100_2$

Convert 43_{10} from decimal to hex. $\rightarrow 101011_2 \rightarrow 2B_{16}$

Convert 43_{16} from hex to decimal. $\rightarrow 1000011_2 \rightarrow 67_{10}$

Convert 1010110_2 from binary to octal.

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex. $\rightarrow 5C_{16}$

Convert AC_{16} from hex to binary. $\rightarrow 10101100_2$

Convert 43_{10} from decimal to hex. $\rightarrow 101011_2 \rightarrow 2B_{16}$

Convert 43_{16} from hex to decimal. $\rightarrow 1000011_2 \rightarrow 67_{10}$

Convert 1010110_2 from binary to octal. $\rightarrow 126_8$

Convert 75_8 from octal to binary.



Practice: Bin, Dec, Hex, Octal

Convert 101_2 from binary to decimal. $\rightarrow 5_{10}$

Convert 101_{10} from decimal to binary. $\rightarrow 1100101_2$

Convert 1011100_2 from binary to hex. $\rightarrow 5C_{16}$

Convert AC_{16} from hex to binary. $\rightarrow 10101100_2$

Convert 43_{10} from decimal to hex. $\rightarrow 101011_2 \rightarrow 2B_{16}$

Convert 43_{16} from hex to decimal. $\rightarrow 1000011_2 \rightarrow 67_{10}$

Convert 1010110_2 from binary to octal. $\rightarrow 126_8$

Convert 75_8 from octal to binary. $\rightarrow 111101_2$



AND OR

10110_2 **AND** 101_2

23_8 **OR** 11_8

20_{10} **XOR** 26_{10}



AND OR

$$10110_2 \text{ AND } 101_2 \rightarrow 00100_2$$

$$23_8 \text{ OR } 11_8$$

$$20_{10} \text{ XOR } 26_{10}$$



AND OR

$$10110_2 \text{ AND } 101_2 \rightarrow 00100_2$$

$$23_8 \text{ OR } 11_8 \\ \rightarrow 010011_2 \text{ OR } 001001_2 \rightarrow 011011_2 \rightarrow 22_8$$

$$20_{10} \text{ XOR } 26_{10}$$



AND OR

$$10110_2 \text{ **AND** } 101_2 \rightarrow 00100_2$$

$$23_8 \text{ **OR** } 11_8 \\ \rightarrow 010011_2 \text{ OR } 001001_2 \rightarrow 011011_2 \rightarrow 22_8$$

$$20_{10} \text{ **XOR** } 26_{10} \\ \rightarrow 10100_2 \text{ XOR } 11010_2 \rightarrow 01110_2 \rightarrow 14_{10}$$



Practice: Domain and Range

sassy length of sass contains sassier not not sassiest

What is the domain of sassy?

Domain of sassier?

Range of sassy?

Range of sassier?



Practice: Domain and Range

sassy length of sass contains sassier not not sassiest

What is the domain of sassy? integers (non-negative)

Domain of sassier?

Range of sassy?

Range of sassier?



Practice: Domain and Range

sassy length of sass contains sassier not not sassiest

What is the domain of sassy? integers (non-negative)

Domain of sassier? boolean

Range of sassy?

Range of sassier?



Practice: Domain and Range

sassy length of sass contains sassier not not sassiest

What is the domain of sassy? integers (non-negative)

Domain of sassier? boolean

Range of sassy? list

Range of sassier?



Practice: Domain and Range

sassy length of sass contains sassier not not sassiest

What is the domain of sassy? integers (non-negative)

Domain of sassier? boolean

Range of sassy? list

Range of sassier? anything

Practice: Domain and Range

From Fall 2011:

Question 6a : The function `foo` is used in the following way (and you have no idea what `a`, `b`, or `c` are). What can you say about the *domain* and *range* of `foo`?



Domain of `foo` (first argument):

Domain of `foo` (second argument):

Range of `foo`:

Practice: Domain and Range

From Fall 2011:

Question 6a : The function `foo` is used in the following way (and you have no idea what `a`, `b`, or `c` are). What can you say about the *domain* and *range* of `foo`?



Domain of `foo` (first argument):

Domain of `foo` (second argument):

Range of `foo`:

1. boolean (true/false)

Practice: Domain and Range

From Fall 2011:

Question 6a : The function `foo` is used in the following way (and you have no idea what `a`, `b`, or `c` are). What can you say about the *domain* and *range* of `foo`?



Domain of `foo` (first argument):

Domain of `foo` (second argument):

Range of `foo`:

1. boolean (true/false), 2. anything

Practice: Domain and Range

From Fall 2011:

Question 6a : The function `foo` is used in the following way (and you have no idea what `a`, `b`, or `c` are). What can you say about the *domain* and *range* of `foo`?



Domain of `foo` (first argument):

Domain of `foo` (second argument):

Range of `foo`:

1. boolean (true/false), 2. anything 3. boolean (true/false)



Practice Problem: Debugging

This script is supposed to allow this sprite to play Rock, Paper, Scissors 3 times in a row against another sprite (not in this picture). What's wrong with it?





Practice Problem: Debugging

Bug 1: “choice” increases three times by a number between 1 and 10, but the “Rock Paper Scissors” list is only 3 items long! What’s item 4 of a 3-item list...?



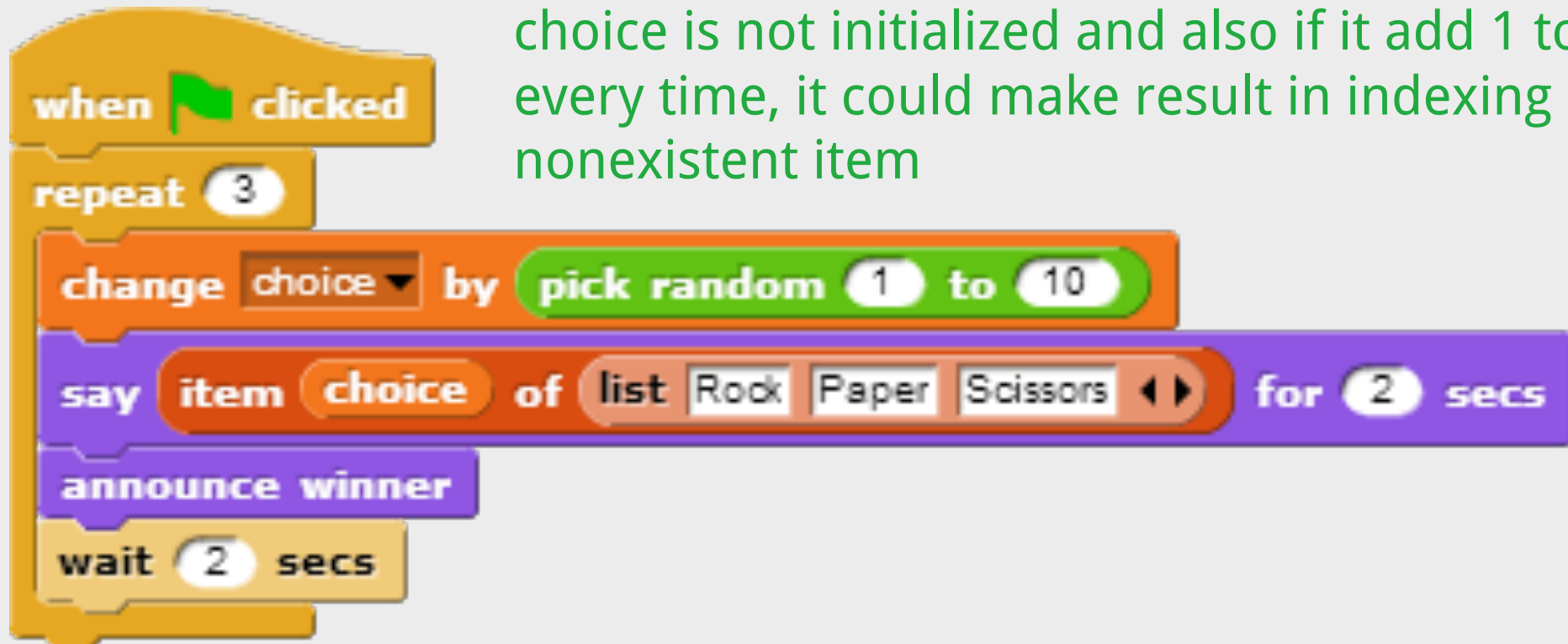


Practice Problem: Debugging

Bug 1: "choice" increases three times by a number between 1 and 10, but the "Rock Paper Scissors" list is only 3 items long! What's item 4 of a 3-item list...?

Bug 2: "change" -> "set"

choice is not initialized and also if it add 1 to 3 every time, it could make result in indexing a nonexistent item

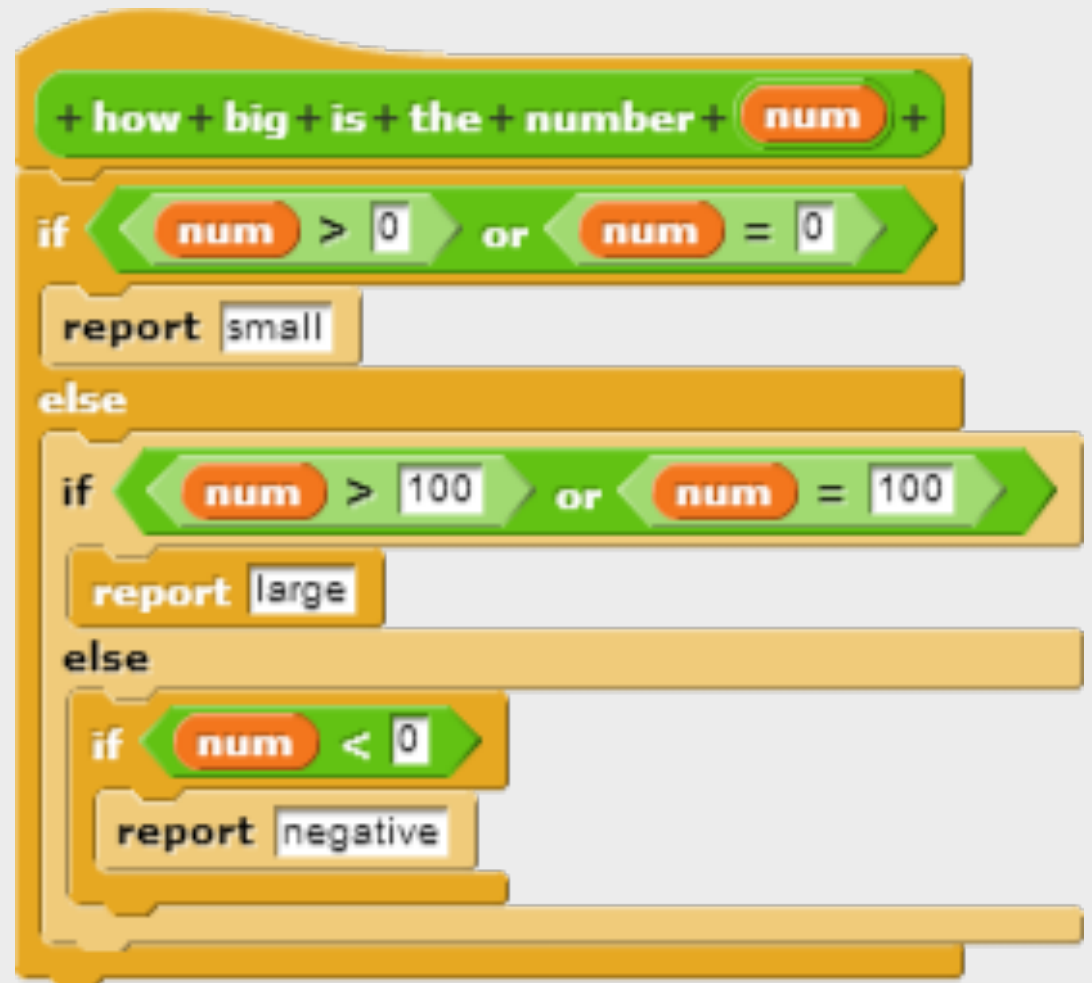




Practice Problem: Debugging

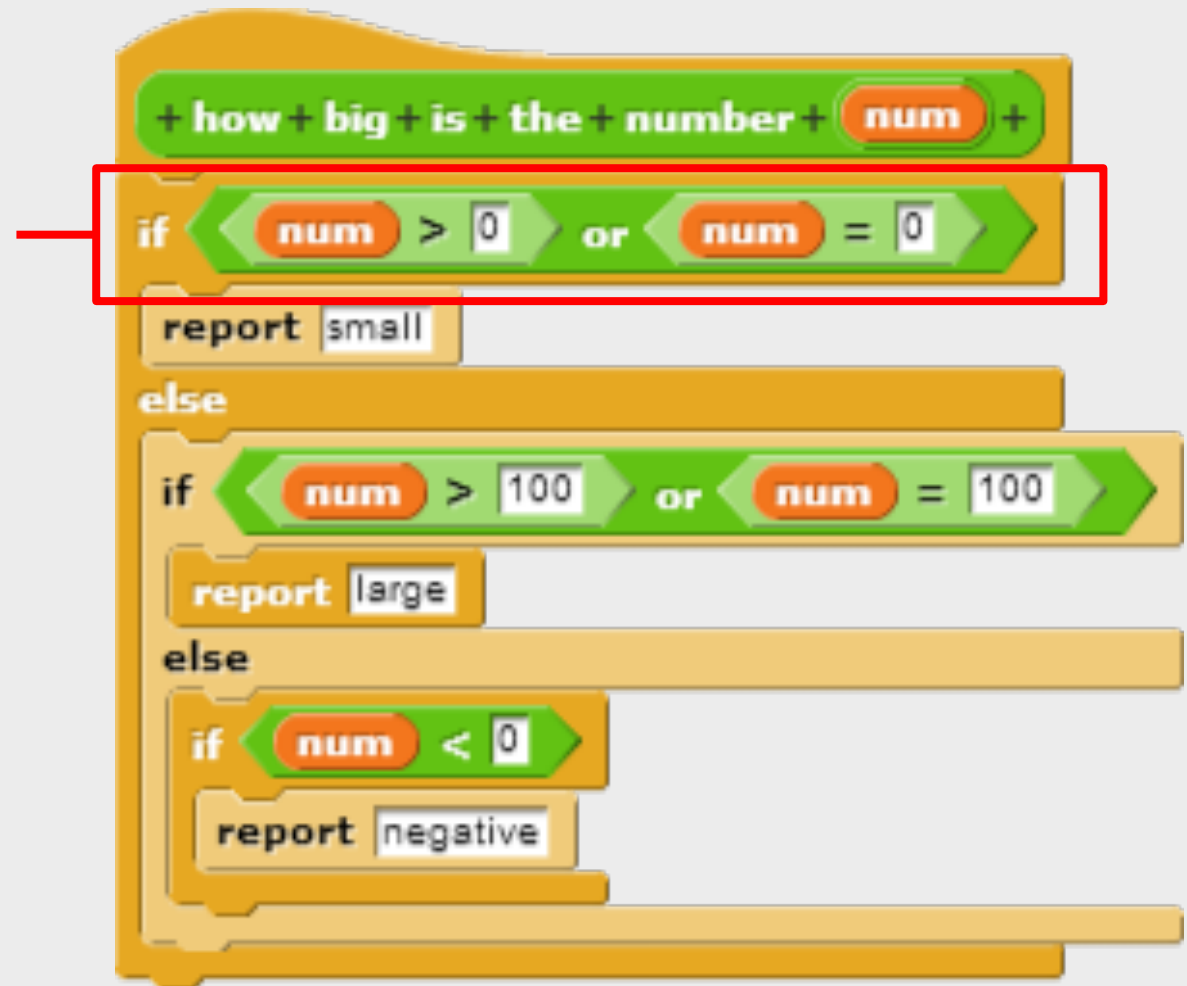
See if you can find the bug in this block. This is the behavior that the block is meant to have:

- Reports "negative" when $\text{num} < 0$
- Reports "small" when $0 \leq \text{num} < 100$
- Reports "large" when $\text{num} \geq 100$





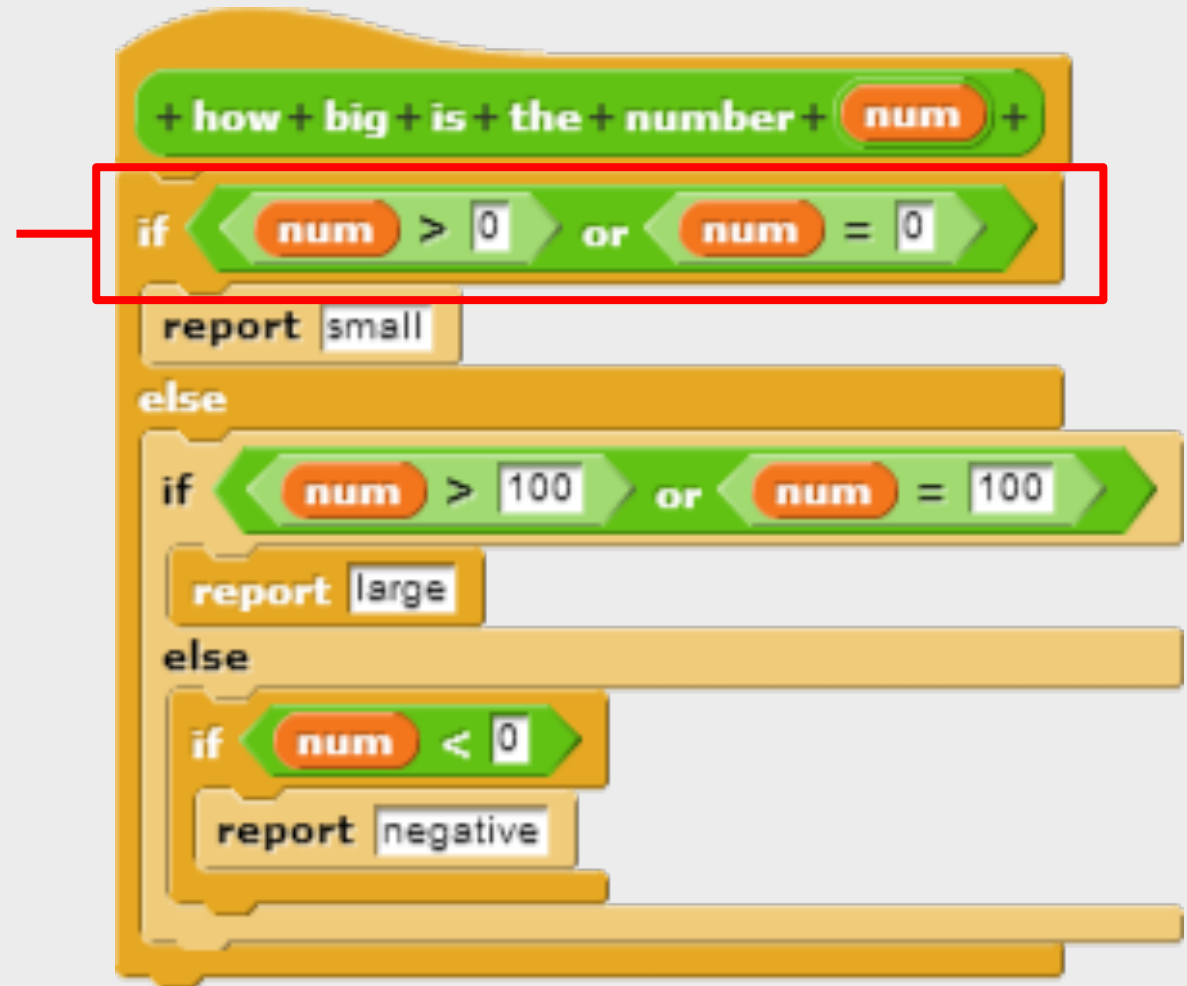
Practice Problem: Debugging





Practice Problem: Debugging

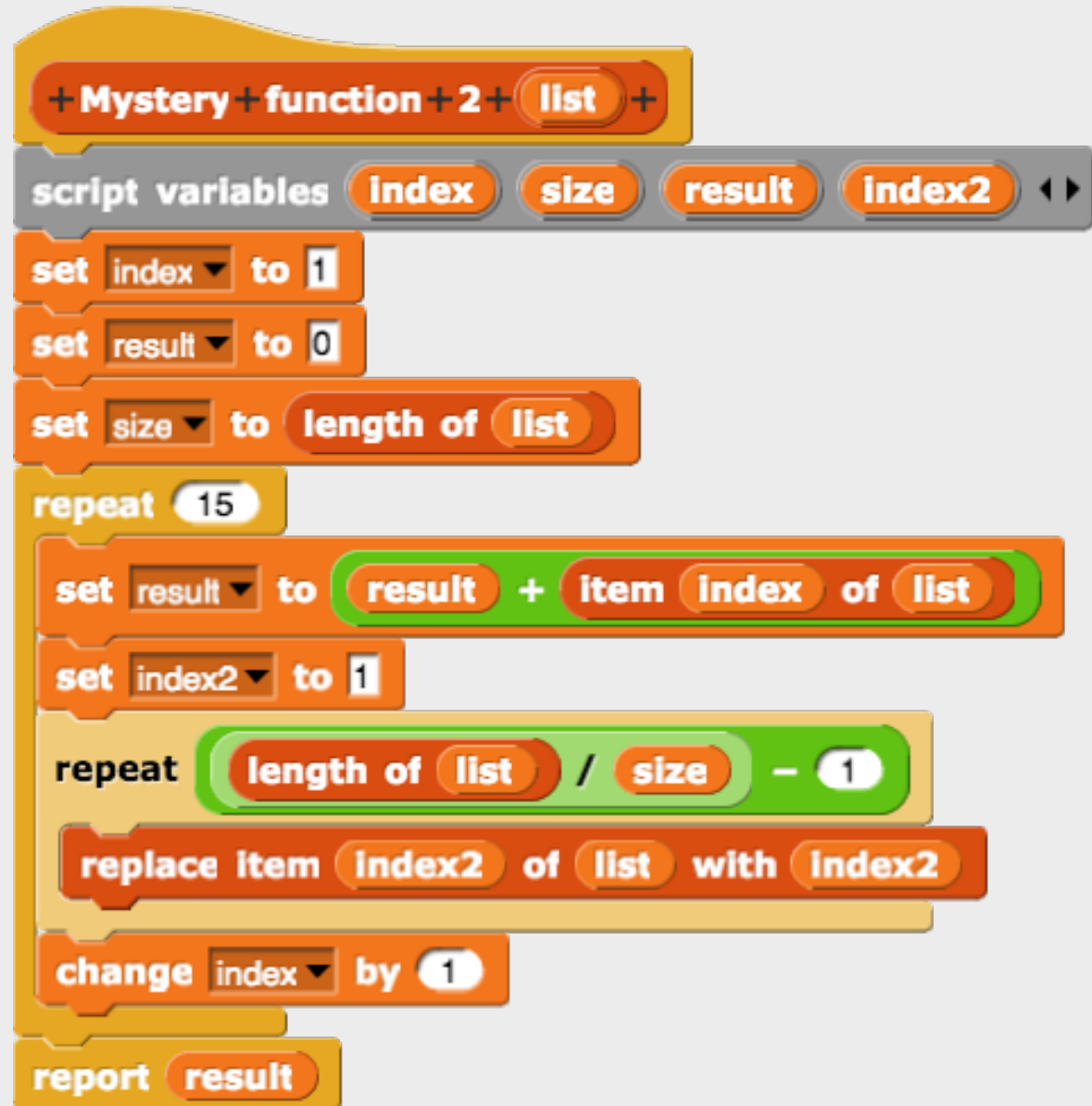
All numbers ≥ 0 will report "small."



Practice: What does it do?

From the
Algorithmic
Complexity
discussion slides:

1. *What does this block do and what does it report?*
2. *What is the running time of this block?*



Practice: What does it do?

What does this block do and what does it report?

sums up the first 15 items of a list

What is the running time of this block?



Practice: What does it do?

What does this block do and what does it report?

sums up the first 15 items of a list

What is the running time of this block?

constant: $O(1)$



Practice: What does it do?

What does this block report?



The image shows a Scratch code block with the following structure:

- combine with** (green flag icon) **+** **items of**
- keep items such that** (green flag icon) **mod** **3** **<** **2** **from**
- map** (green flag icon) **length of** (empty input field) **over** **list** **234** **8765432** **32** **45678** **123456789** **2** (with left and right arrow buttons)

Practice: What does it do?



report 20

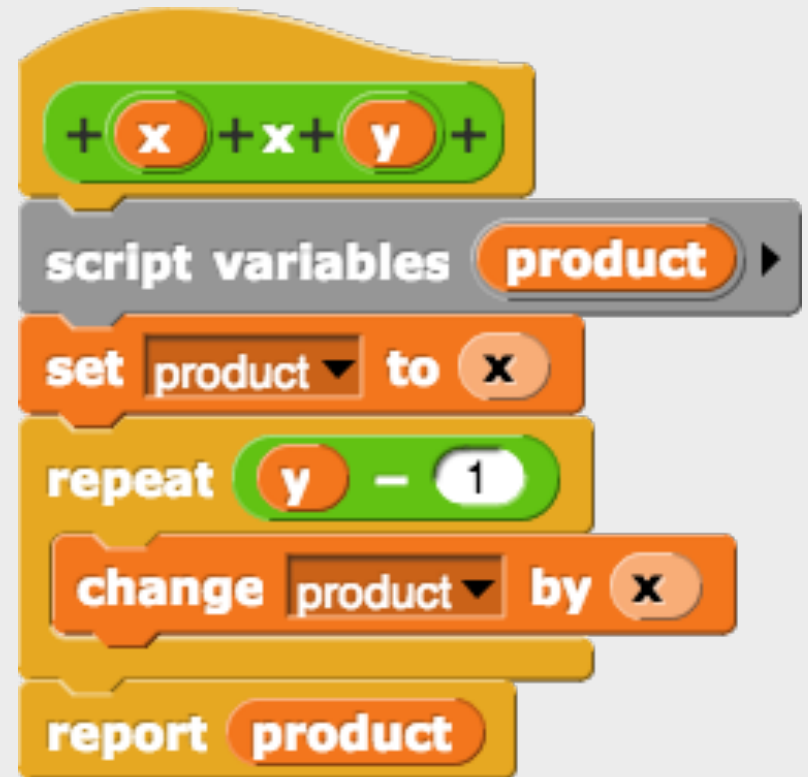
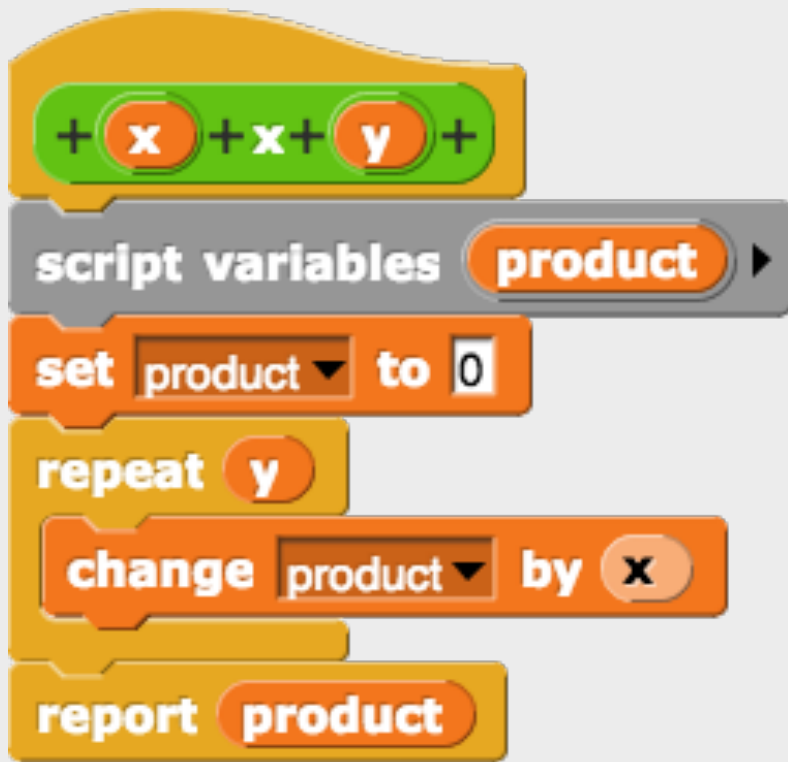


Practice Problem: Make a Block

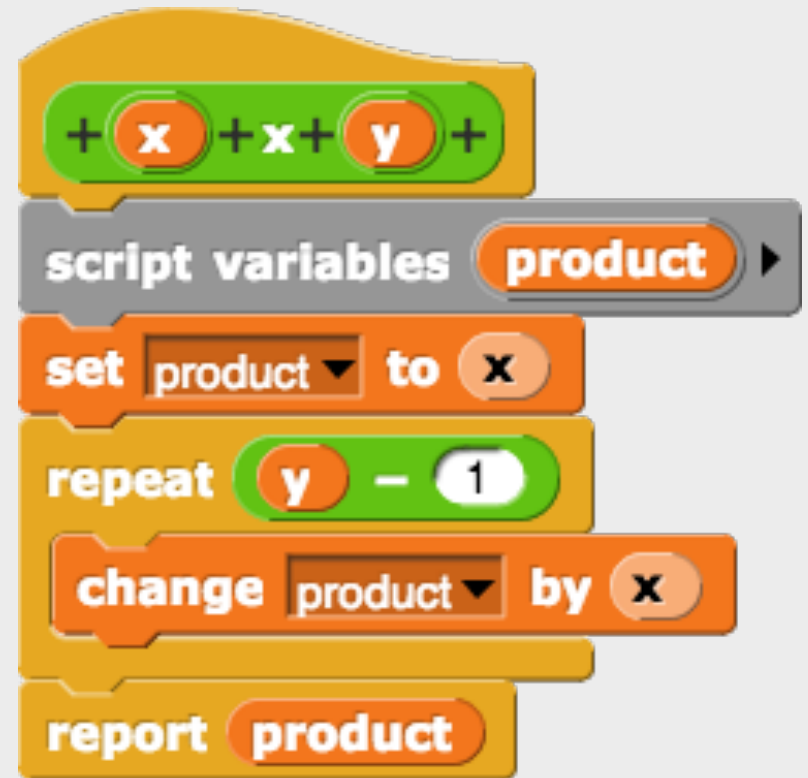
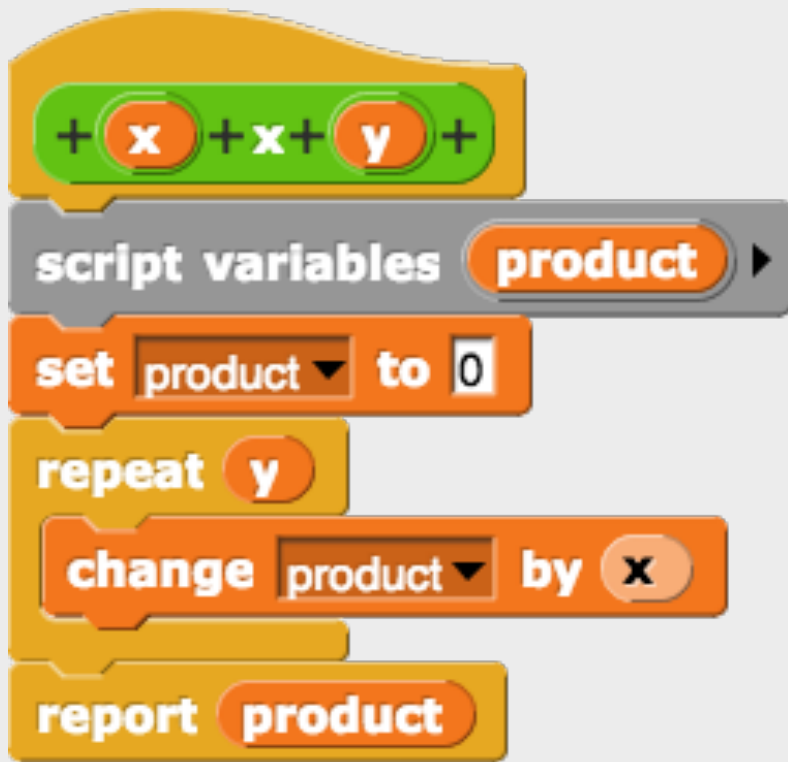
Write a new version of multiply called "mult (x) (y)". You are not allowed to use the $() * ()$ block. You also are not allowed to use the $() / ()$ block. (*hint: consider using a loop*)

What is the run time for this block?

Mult (x) (y)



Mult (x) (y)



Run time: $O(n)$



Practice Problem: Make a Block

Using HOFs, create a block that deletes all the elements of a list that end in the digit x (where x is an input to the block).

Delete All Items...

+delete+all+items+from+ **list** +that+end+with+ **x** +

report

keep items such that **not** **letter** **length of** **of** **=** **x** **from**
list



Extra Practice Problems

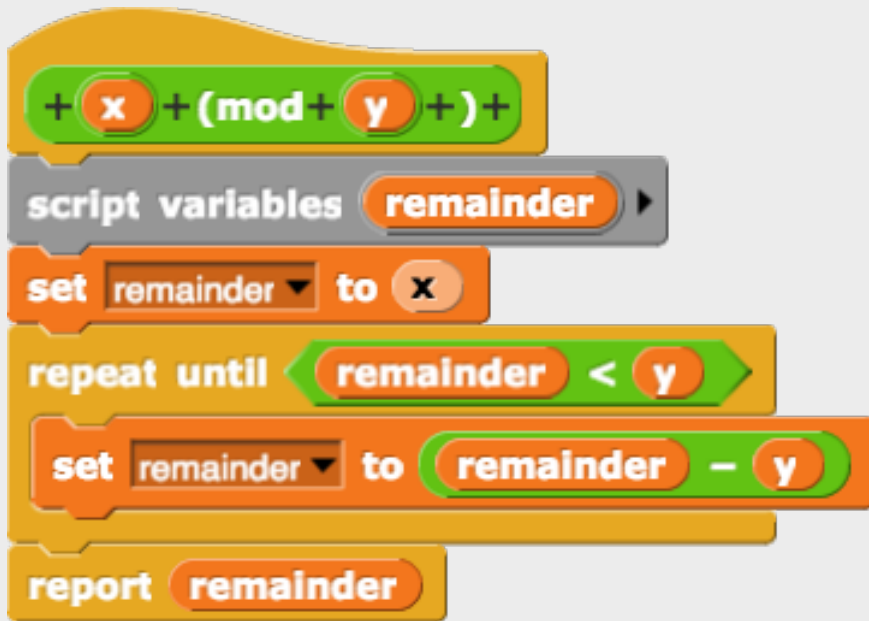
You can try to make these with or without HOFs.
If you don't use HOFs, note the runtime in your solution.



Practice Problem: Extra

The function `mod` takes a number `x` and a number `n` and reports the remainder of dividing `x` by `n` (when `n > 0`). Write a new "`mod (x) (n)`" block that works for positive values of `n`. You are not allowed to use the `() mod ()` block.

Mod (x) (n)



Does not work for negatives.



Works for negatives.



Practice: Extra

Count the number of times the first element of list repeats throughout the list.

Count the Number of Times...

Solution 1:



Solution 2:





Practice Problem: Extra

Write a block that takes a list as a parameter and reports a new list where the order of the elements in the list is reversed.

Reverse List





Practice Problem: Extra

Create a block "count the letter (L) in the list (list)" that takes a letter as input and counts the number of times this letter occurs in the entire list. In other words, this is the number of times the letter occurs in word 1, plus the number of times the letter occurs in word 2, and so on.

Try this problem two ways:

- 1) Create this block using HOFs (*hint: make a helper block also using HOFs*)
- 2) Create this block without HOFs (*What is the order of growth for this block?*)

Count the Letter... (with HOFs)





More Resources

Links have been put on the course website for extra quest resources and past discussions.

Quest Resources:

<http://cs10.org/sp15/exams/quest.html>

Discussion 3: Lists

<http://inst.eecs.berkeley.edu/~cs10/sp14/disc/03/>

Discussion 4: Algorithmic Complexity and Quest Review

<http://inst.eecs.berkeley.edu/~cs10/sp14/disc/04/>