

CS10 Final Exam Review

August 5, 2012

Slides are available **NOW!**

<http://bit.ly/OE3EL4>

First up: administrative things

Second: readings

Third: programming

Time and Place

The final exam is being held on
Wednesday, August 8
from 6 - 9 pm.

We will be in
2060 VLSB.

Other Stuff to Know

The exam is **cumulative**.

You will be allowed to bring **THREE** sheets of paper.
Double-Sided, handwritten only

You can also have the Blocks sheet which you used on the online final

Get there a little bit early to ensure that you find the room before the test starts.

BRING YOUR CLICKERS!

(If you haven't already. :))

What to Expect

Valid stuff for us to cover include:

- Any of the readings (including required videos).
- Any of the labs except for simulations.
- Any of the lectures (including Guest lectures).

Readings

Week 1

Why Software is Eating the World (CS & Society)

- Many companies that are thriving today are the technology-based counterparts of old industries.
 - Netflix <== Blockbuster
 - Amazon <== Borders
 - iTunes, Spotify, Pandora <== music stores
 - Shutterfly, Flickr <== Kodak
- Some of the most valuable companies in the world are tech companies (Microsoft, Google, Apple).

Justices Split on Violent Video Games

- The rise of new technology often leads to the need for new legislation. This article is a bit dated, but describes a conflict that the Supreme Court dealt with a year or so ago: whether states can prohibit the sale of certain games to minors.

Games with a Purpose

- A game that generates computationally useful output when played.

Week 1 (continued)

Intro to Abstraction

- Abstraction is one of the fundamental ideas of the field of computing. Learning to master it is challenging but incredibly powerful.
- One key idea of abstraction is creating an "interface:" a way for outside actors to use our systems. The interface to a car's engine consists of a gas pedal, brake pedal, and in some cases a clutch and gear shift. The interface to an air conditioning involves "increase temperature" and "decrease temperature" buttons.
- An interface to a piece of software is often a set of functions that are intended to be used to perform some certain behavior. They often hide certain details of the implementation from the user.

Week 2

Program or Be Programmed (CS & Society)

- Those who can program ("creating" in the digital age) are going to be the individuals who have the control, similar to printers, scribes, and priests of the past.
- It is critical that the public learns not only how to use machines in the ways that others tell us we can, but to learn how to use them however we want to.
- Ten commandments for introductory-level understanding.

Algorithms & Algorithmic Complexity

- TED Talks: How Algorithms Shape Our World
- What is an algorithm?
- How do we measure an algorithm's performance?
 - Constant
 - Logarithmic
 - Linear
 - Quadratic
 - Cubic
 - Exponential

Week 2 (cont.): Blown to Bits, Ch. 1

Some of the biggest social shockwaves from the computer revolution have come from the presence of digital information.

Digital information includes anything that is stored electronically as a series of bits (electronic books, emails, websites, Facebook profiles, digital photos, your middle school chat logs, electronic medical records, etc).

A bit is the smallest chunk of digital information that we recognize. Bits (digital information) are novel because:

- They can be copied perfectly.
- They can be transmitted instantly.
- They can be stored in small places.
- You can write programs to deal with them (search, etc).

Week 2 (cont.): Blown to Bits, Ch. 2

- Privacy Lost, Privacy Abandoned
- Footprints & Fingerprints
- RFIDs
- Why We Lost Our Privacy, or Gave it Away
- Privacy & Freedom
- Fair Information Practice Principles

Week 2 (cont.): Programming Paradigms

The four primary paradigms:

- Functional
- Imperative
- Object-Oriented
- Declarative

All are equally powerful! All are Turing complete! Huzzah!

Week 3: Free Lunch is Over

(Concurrency) Until recently, programmers could count on getting a "free" speedup for their software as hardware continued to increase in speed.

Nowadays this isn't necessarily the case because we're forced to develop applications that take advantage of **multiple cores** instead of just a single faster one. This means we need to take concurrent programming concerns, such as resource sharing, "live-lock," and "dead-lock," into account to get maximum performance.

Week 3 (cont.): How Moore's Law Works

- Gordon E. Moore (Intel co-founder, and UC Berkeley Grad! Woot!): The number of transistors to integrated circuits doubles per unit of space approximately every 18 to 24 months

Week 4: Search Engines in a Nutshell (Blown to Bits, Ch. 4)

The power of today's search engines compared with technology a decade ago is hard to describe. Many new things are possible.

Phase 1 - Build the Index

1. Gather information. Pull in as much data as possible.
2. Keep copies. Keep it around for processing purposes.
3. Build an index. Create a structure that allows you to find information quickly.

Phase 2 - Search the Index

4. Understand the query. Analyze what the searcher is looking for.
5. Determine relevance. Filter out information that's unrelated to the user's search.
6. Determine ranking. Organize the relevant stuff by HOW relevant it is.
7. Present the information. Send back the results.

Week 5: More Implications

- Applications that Changed the World
- BtB Chapter 5
 - Cryptography
 - Public Key Encryption
- BtB Chapter 6
 - Copyright

Week 6: Blown to Bits, Ch. 7

- Guarding the Frontiers of the Digital Expression
- Regulating the internet
- Communications Decency Act
- Digital Protection & Digital Censorship

Week 6 (cont.): The End Is Near, Maybe

The change is accelerating as more becomes possible!

- **Personal identity?** The internet cannot guarantee it.
- **Free speech.** Every word you say online is permanent.
- **Creativity.** Protecting an individual's ideas are difficult in the digital landscape.

Week 6 (cont.): Life in the Cloud

Cloud computing = Internet-based computing

Cloud computing is a growing trend in business and research. It offers a number of advantages over building your own data center, including lower start-up costs, seemingly infinite available resources, and the option of paying for short-term usage.

Top 10 Challenges for Cloud Computing (in article)

Cloud computing also has privacy implications since it involves storing information or documents with a single organization (Amazon, Google, Microsoft, etc).

Week 6 (cont.):

HoFs + Distributed Computing

- Higher Order Functions (Lambdas)
 - Map
 - Keep
 - Combine
 - ... and LOTS of others! Not just these three!
 - Make your own HoFs!
- What is distributed computing?
- MapReduce (More than just Map + Reduce)

Week 7

- Brian Harvey's Artificial Intelligence Notes
 - John McCarthy's definition of AI
 - "Getting a computer to do things which, when done by people, are said to involve intelligence"
- Game Theory
 - Strongly and weakly solving a game

Week 7 (cont.): Artificial Intelligence

IBM's **Watson** supercomputer was designed to answer questions posed in normal language?

The **DARPA Challenge** described in "The Great Robot Race" was a challenge issued by the US Government to build a robot that could navigate on its own?

A little place called Stanford won the DARPA Challenge on the 2nd year of the competition.

After 18 years of performing computations, **checkers is now officially solved**. If no mistakes are made it results in a draw.

Week 7 (cont.): Quantum Leap

Today's technology is child's play compared to the promises of quantum computing.

Quantum computing offers the promise of higher speeds and smaller form factors: the articles talk about computers in paint, furniture, in our bodies, etc. Power requirements will be minimal compared to today's technology.

Instead of bits, quantum computing takes advantage of something called 'qubits.' They are very weird and you don't need to understand the physics behind them for this exam.

Weeks 5, 7: Lectures

- Saving the World with Computing (Kathy Yelick)
- HCI (Bjoern Hartmann)
- AI (Anna Rafferty)
- Limits of Computing
- Future of Computing

Reading Q&A

***Anything* in the Class**

Programming

(concept poll)

Recursion: ROW ROW FIGHT DA POWAH

Write the following block recursively:

 raised to 

Recursion: ROW ROW FIGHT DA POWAH

Write the following block recursively:



Recursion: Pattern Matcher

Write the block



that determines if the first input (a word) matches a pattern that is given as the second/ input. The * character is a *wildcard*: it matches *any* character. (You may find some of the blocks in the ToolSprite useful.)

true

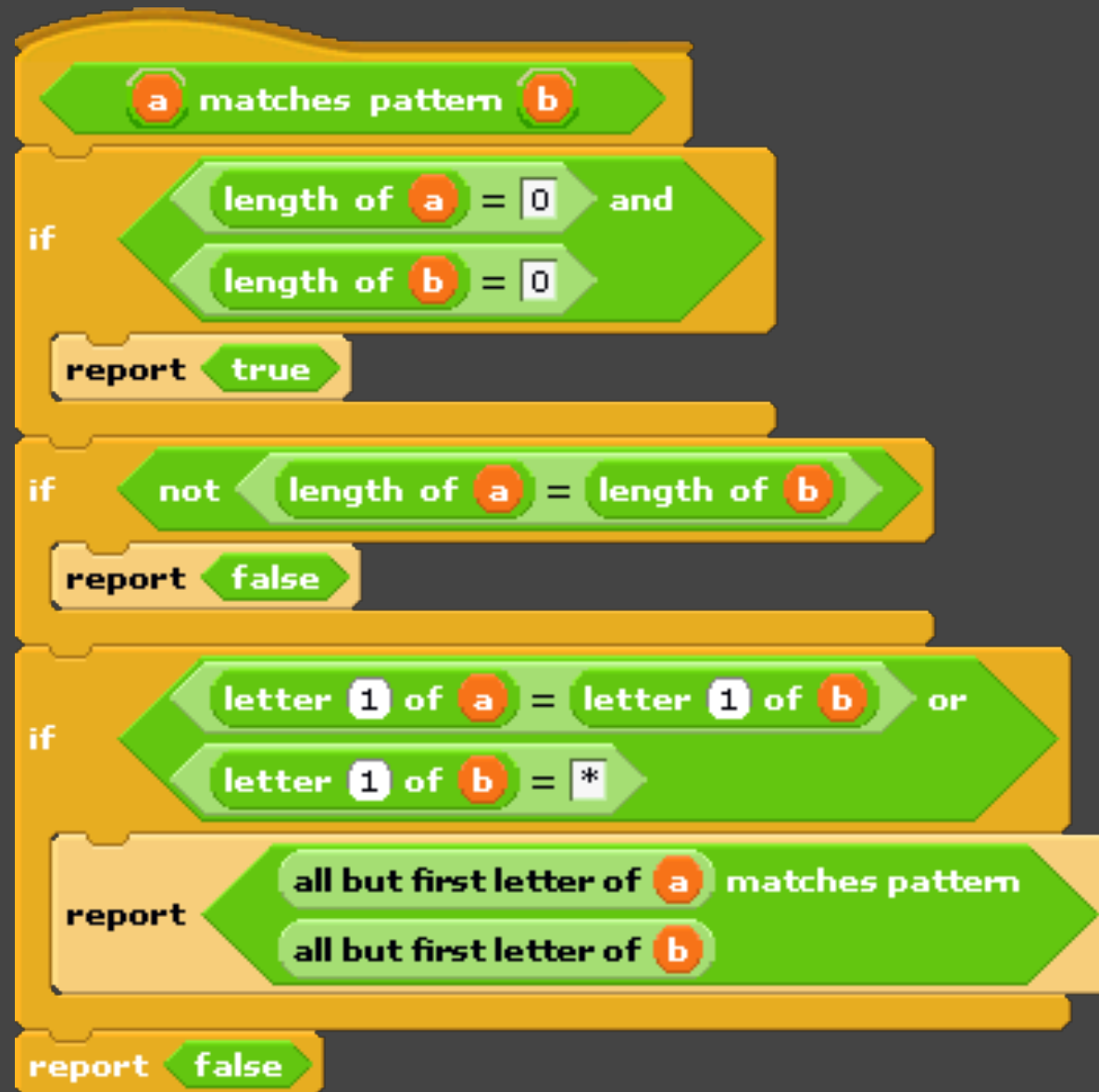
true

false

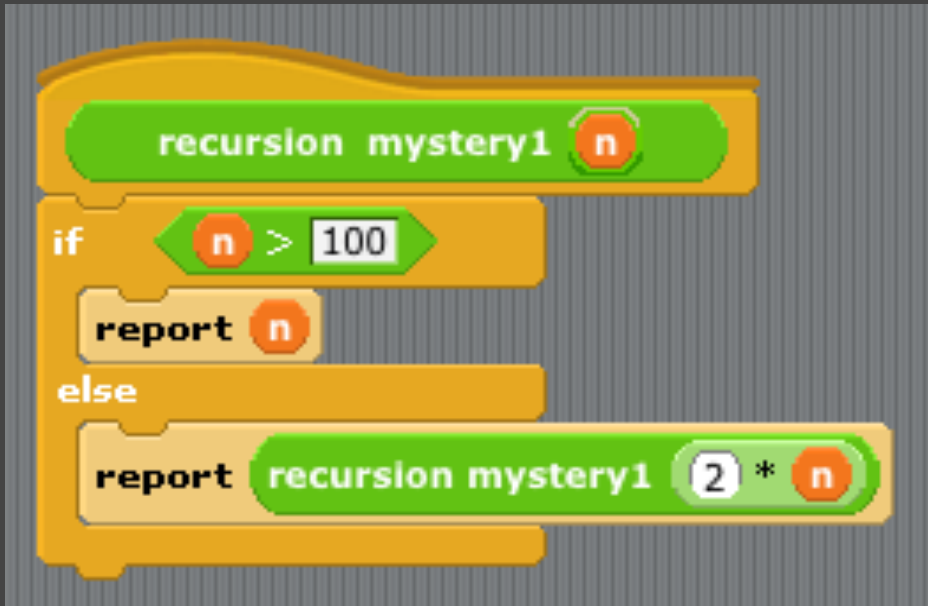
false

Recursion: Pattern Matcher

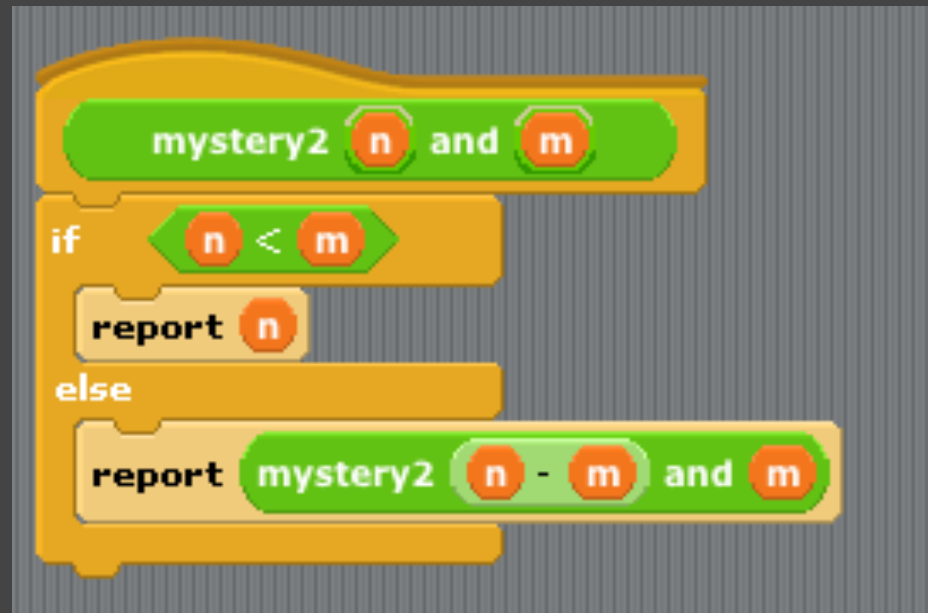
matches pattern



Recursion Mystery



What does mystery1 report when $n = 113$? 70? 42? 30? 10? 9999?

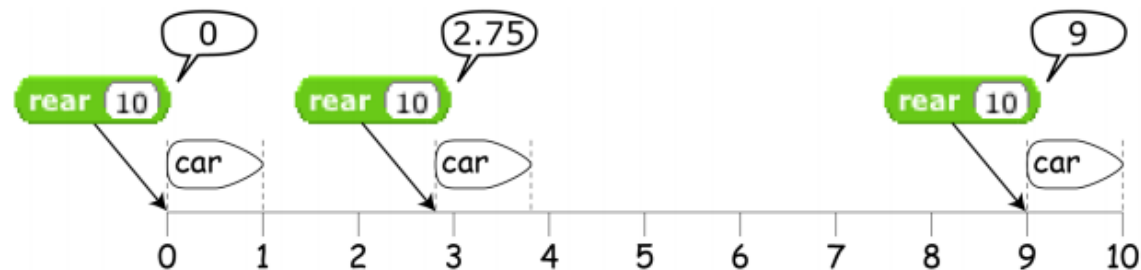


What does mystery2 report when $n, m = 6, 13$; 14, 10; 37, 10; 8, 2; 50, 7?

Question 12: *Finding Parking in Berkeley. Get it?*

Have you ever tried to park on a side street in a Berkeley residential area? People do such a poor job parking that they waste most of the street. How many 1-unit-long cars can park on a street given that people park randomly? Let's simulate it! We're going to assume cars don't need any extra "breathing room" between them to park and can just "drop" into a tight space.

Luckily, someone else has provide a helper block called `rear space`, which takes in the amount of space left, and *randomly* picks a place for the car to park in that space, reporting *the location of the car's rear end* (hence the name). It is an error to call `rear space` with less than 1 unit of free space available. As an example, a call to `rear 10` would return a number from 0 (car parked in the *back* of the 10-unit spot) through 9 (car parked at the *front* of the 10-unit spot) or any number in-between, like 2.75. All three example reported values are shown below.



- a. In the *best* case, we can get 10 cars to park in a 10-unit space.
How many 1-unit-long cars can park in a 10-unit space in the *worst* case? _____

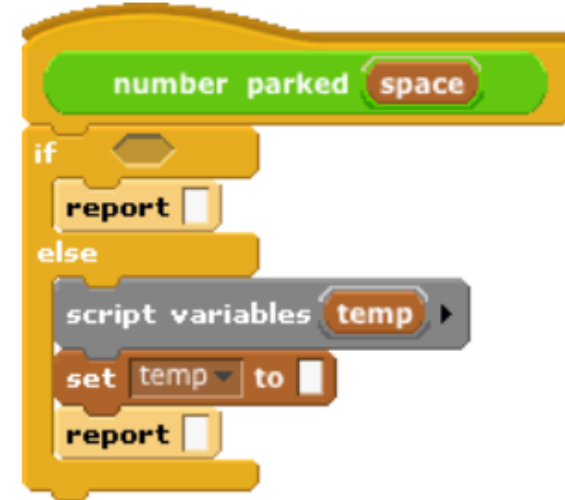
- a. In the *best* case, we can get 10 cars to park in a 10-unit space.
 How many 1-unit-long cars can park in a 10-unit space in the *worst* case? _____
- b. Fill in the blanks to complete a block called `number parked` `space`, which takes in the amount of space in the street, and simulates random parking to estimate the number of cars that are able to park on the street before there's no more space. *Hint: Think of what happens when the first car randomly parks ... it creates two new spaces: front & behind.*

```
number-parked (SPACE)
```

```

if ( _____ )
  report ( _____ )
else
  script-variable (TEMP)
  set- (TEMP) -to- ( _____ )
  report ( _____ )

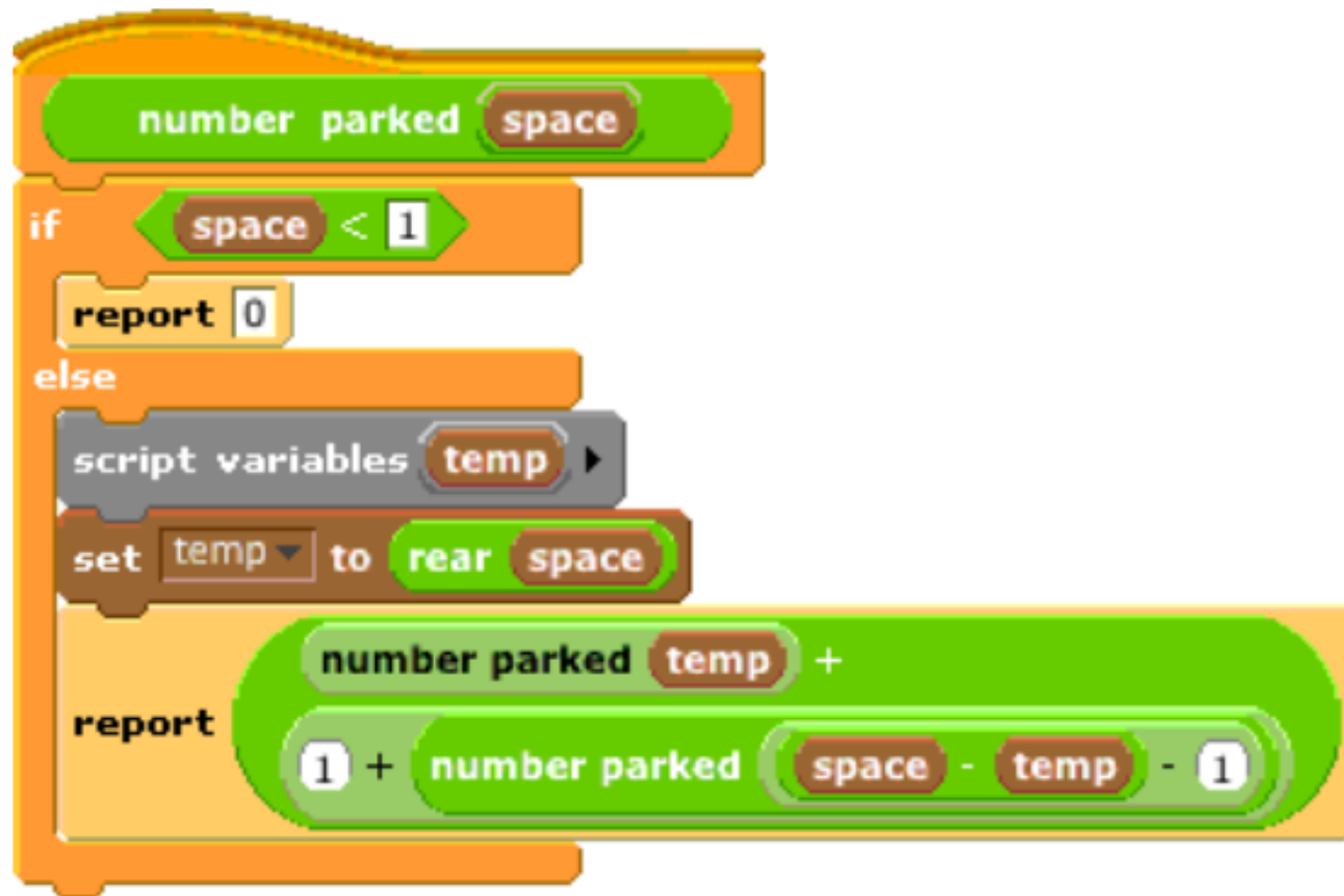
```



Example: `number parked` `space`



Finding a Car in Berkeley: Solution



Adding the Pieces

Write a block that can determine whether an input number can be reached by summing multiple consecutive positive numbers. For example, 43 can be reached by summing $21 + 22$, and 42 can be reached by summing $13 + 14 + 15$.

If it can be reached, report the appropriate sequence of numbers. If not, report an empty list.

*Note: if there are multiple combinations, just report one of them.

Adding the Pieces

one possible solution.

Notes:

Repeat until is same as normal repeat-until but allows for us to store the sum to avoid recalculating.

"Say" blocks are for visualizing the algorithm through Alonzo's mouth.

```
+ summable? num +
script variables start finish sum
set start to 1
set finish to 0
set sum to 0
repeat
  change finish by 1
  say join start join , finish for 0.3 secs
  set sum to sum of numbers from start to finish
until not sum < num
if sum = num
  report list numbers from start to finish
repeat
  change start by 1
  say join start join , finish for 0.3 secs
  set sum to sum of numbers from start to finish
until not sum > num
if sum = num
  report list numbers from start to finish
```

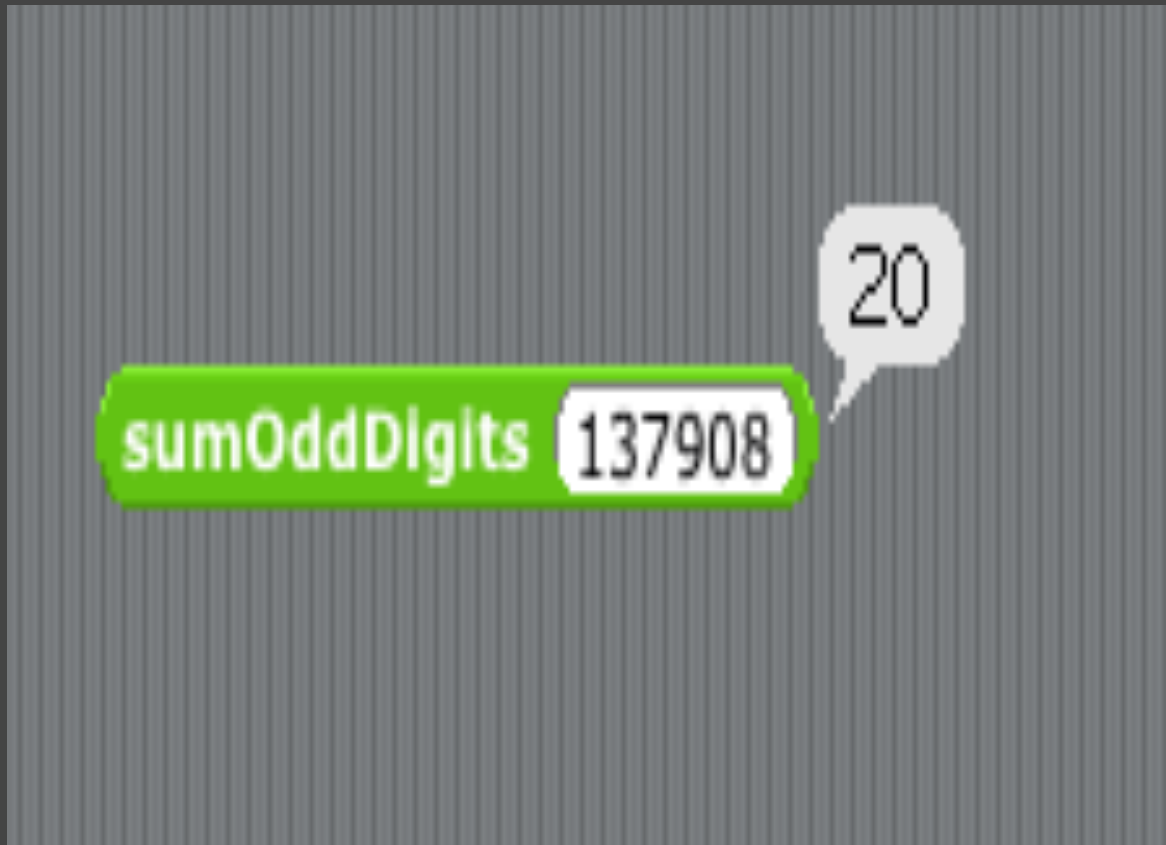
Adding the Pieces

Another solution.

Pv's solution:
has an expanding and
contracting list of
consecutive numbers
adding the next bigger
number to the end when too
small and subtracting the
smallest when too large.



Higher Order Function



Write a block that reports the sum of odd digits in an input by two methods.

In this case:
 $1+3+7+9 = 20$

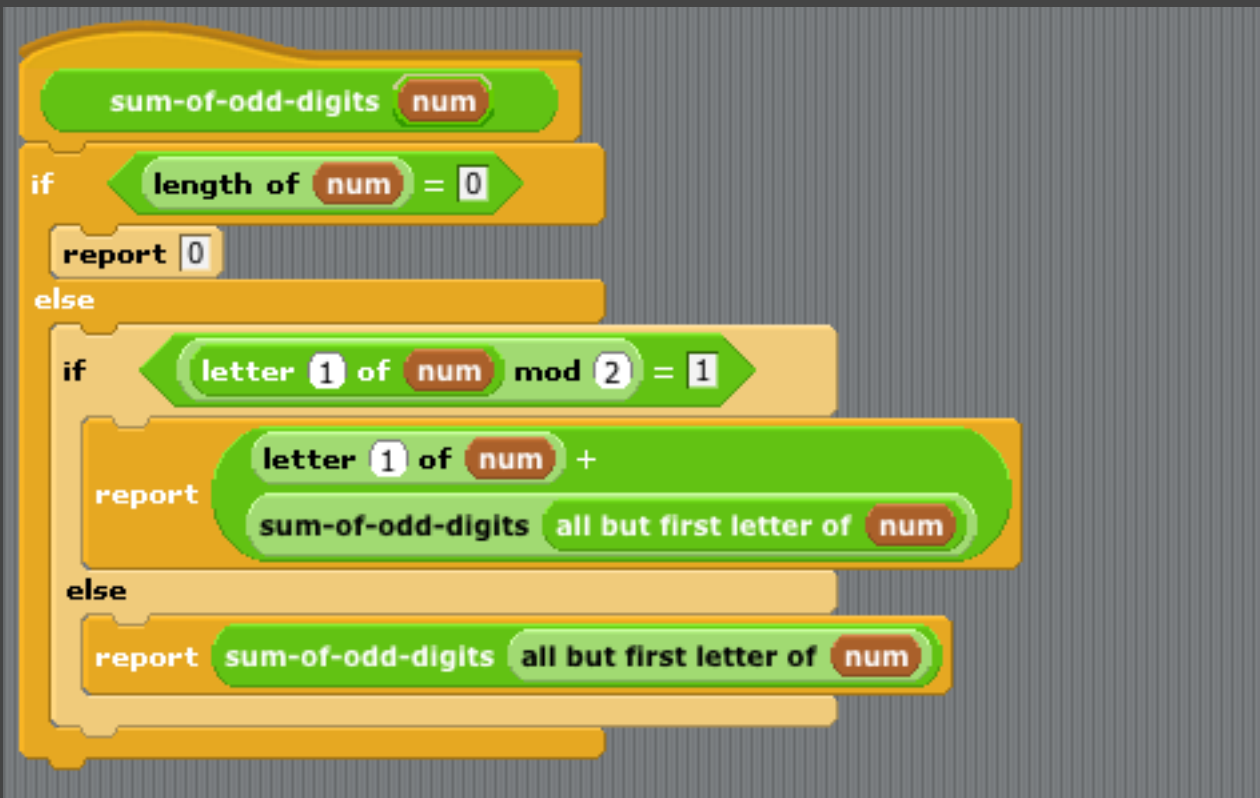
Don't worry about negative values.

Please try to use:
1) HOF only
2) Recursion only

Higher Order Function



Recursive Solution



```
sum-of-odd-digits num
if length of num = 0
  report 0
else
  if letter 1 of num mod 2 = 1
    report letter 1 of num +
      sum-of-odd-digits all but first letter of num
  else
    report sum-of-odd-digits all but first letter of num
```

The image shows a Scratch script for a recursive function named 'sum-of-odd-digits'. The script starts with a function block 'sum-of-odd-digits num'. It then enters an 'if' block with the condition 'length of num = 0'. If true, it reports '0'. If false, it enters an 'else' block. Inside this 'else' block, there is another 'if' block with the condition 'letter 1 of num mod 2 = 1'. If true, it reports the value of 'letter 1 of num' plus the result of a recursive call 'sum-of-odd-digits all but first letter of num'. If false, it reports the result of the recursive call 'sum-of-odd-digits all but first letter of num'.

Higher Order Functions

(a) Write the block

numbers from to

which reports a (new) list of all of the numbers from the first input to the second input. If the first input is larger than the second, the block should return an empty list. (You can use any technique you have learned in this class.)

(b) Write the block

is factorion

which determines if a number is a factorion. A number is a *factorion* if the sum of the factorials of each of its digits is equal to itself. **You should use higher-order functions.**

Higher Order Functions

numbers from to

```
numbers from a to b
script variables index list
set index to a
set list to list
repeat (b - a + 1)
  add index to list
  change index by 1
report list
```

The image shows a Scratch script designed to generate a list of numbers from a given start value 'a' to an end value 'b'. The script begins with a comment 'numbers from a to b'. It then declares two script variables: 'index' and 'list'. The 'index' variable is set to the value 'a', and the 'list' variable is set to an empty list. A 'repeat' loop is used to iterate from 'a' to 'b'. The number of iterations is calculated as $b - a + 1$. Inside the loop, the current value of 'index' is added to the 'list', and 'index' is incremented by 1. Finally, the 'list' variable is reported as the result of the function.

Higher Order Functions

is factortion

is factortion

report

combine with items of

map over

word->list ◀ ▶

=

Higher Order Functions

Write the block

`sublist from position to of`

that takes a starting position, an ending position, and a list, and reports a (new) list that contains all of the elements in the original list from the starting position to the ending position. **Your solution must use only higher-order functions.**

Higher Order Functions

sublist from position to of

sublist from to of

script variables

set to

for each item of

if

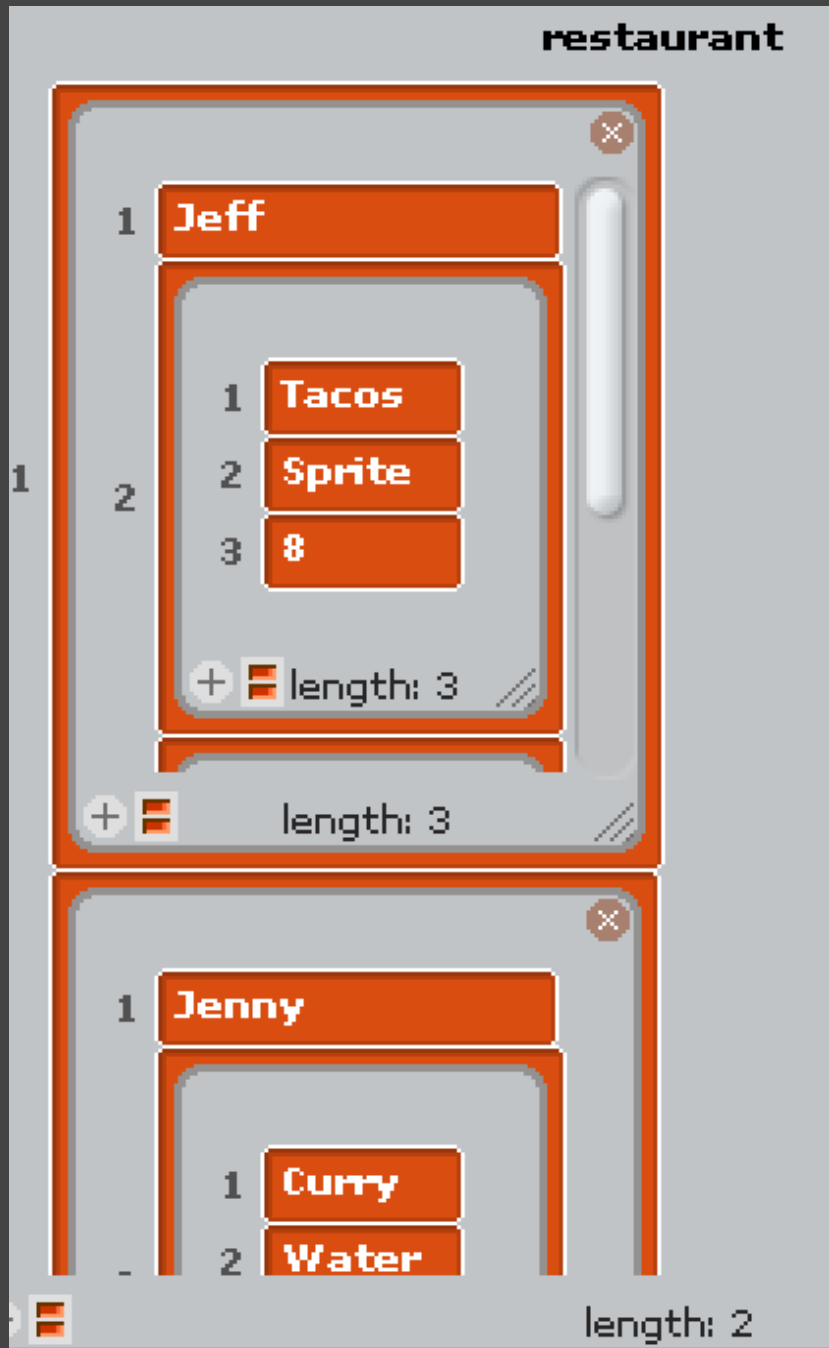
> or = and

< or =

add to

Can also be done with Keep. Try on your own.

Higher Order Function



You own a restaurant and want to implement a new program where you store information on your users preferences. For every customer you want to save their orders including the items in it and the total cost of the order.

Question:

1) How would you add a customer's order if they are a returning customer? If they are new? How would you check?

2) How would you pull the total amount that a particular customer has spent at your restaurant?

3) How would you write a code to determine the most commonly ordered item?

Thought Experiment

What is the most efficient method you can think of for identifying all the unique numbers in a list of 2000?

Describe it, don't write it.

What if you knew all the numbers within 0 to 100?

Questions on the other slides?

- Loops and Variables
- Conditionals
- Lists
- Algorithms and Complexity
- Concurrency
- Recursion
- Data structures
- Lambdas and HOFs