CS10 Final Review

Programming

Concept Review

- Loops & Variables
- Conditionals
- Lists
- Algorithms & Complexity
- Concurrency
- Recursion
- Data Structures
  - ~~Hash Tables~~
- Lamdas & HOFs

Loops & Variables

Multiplying operand1 by operand2

Correct?

Loops & Variables

Multiplying operand1 by operand2

No! Why?

Loops & Variables


```
Multiplying operand1 by operand2
```

Uninitialized Variable

Loops & Variables

Multiplying operand1 by operand2

Initialized Variable!

Loops & Variables

Be sure that your variables are initialized to a correct / known / sane value!

Loops & Variables

Multiplying operand1 by operand2

Correct now?

Loops & Variables

`Multiplying operand1 by operand2`

What if operand1 is negative? Or a "real" number like 2.3?

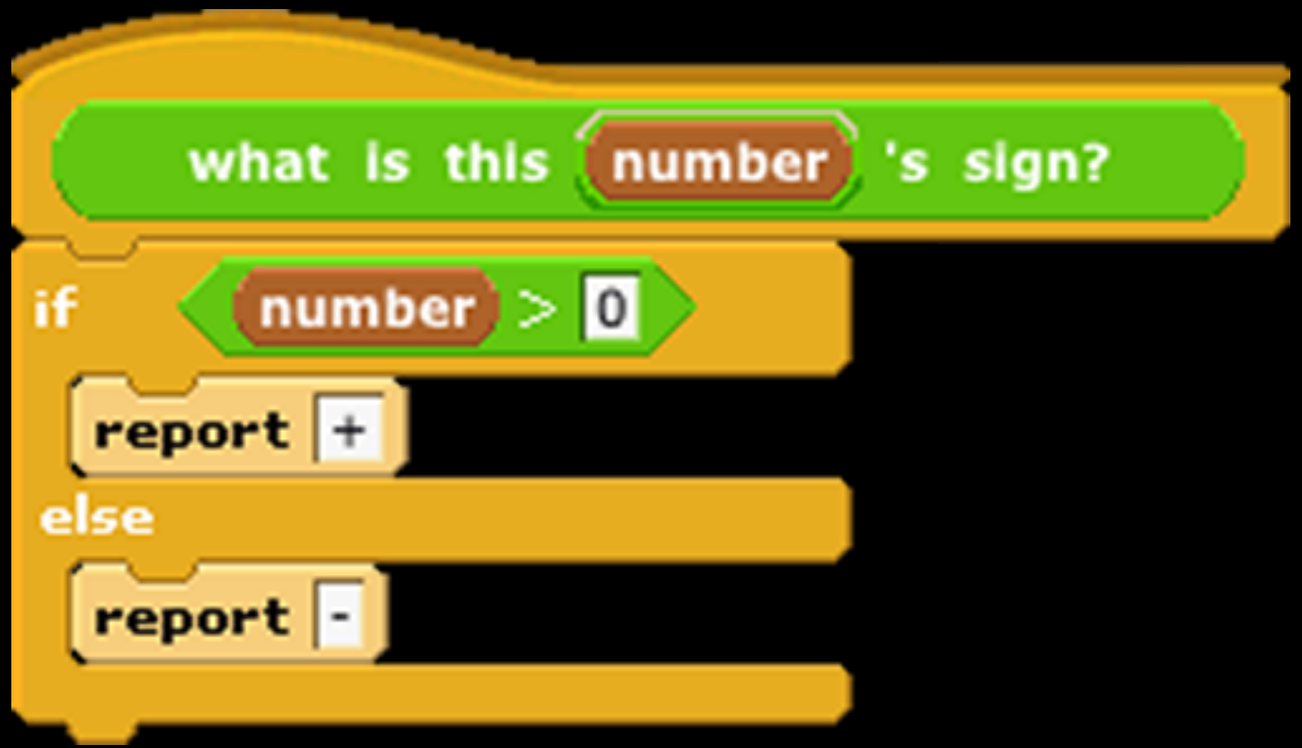
Loops & Variables

Be sure that your looping conditions are valid!

Conditionals

Reports “+” for positive numbers, “-” for negative numbers.

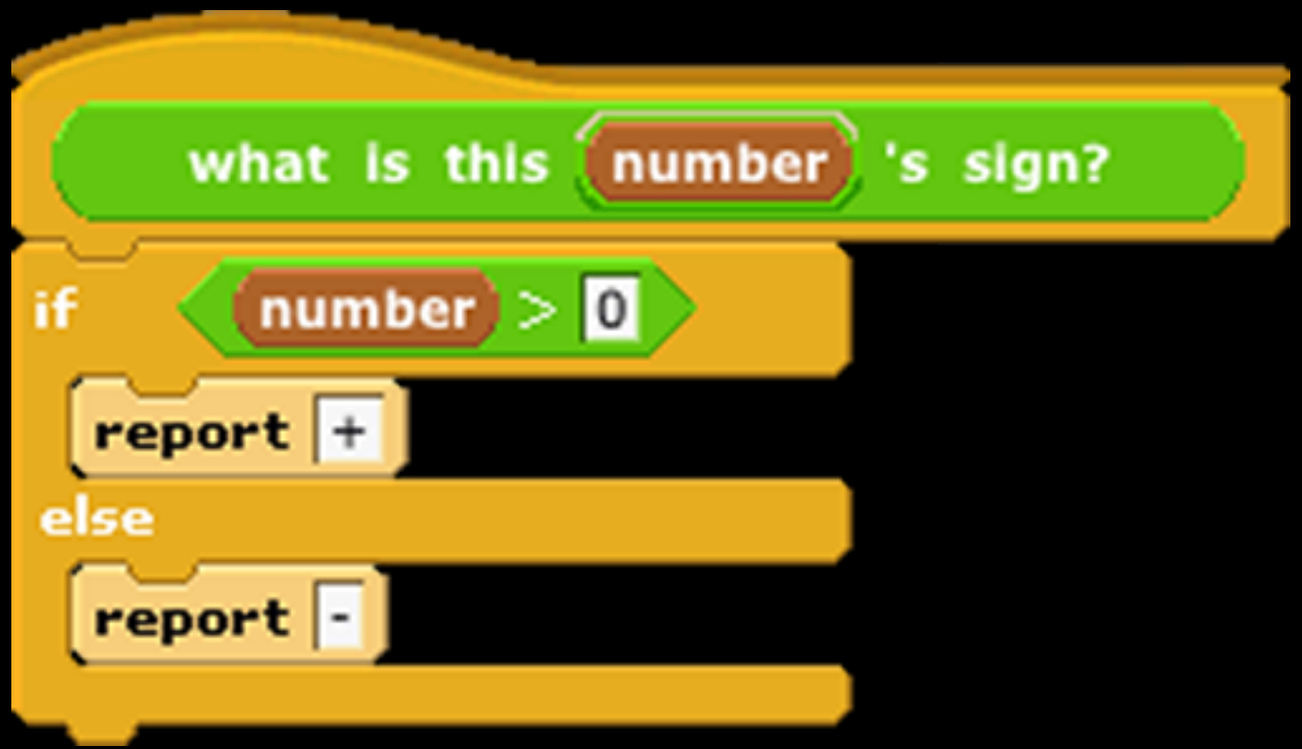
Correct?



Conditionals

Reports “+” for positive numbers, “-” for negative numbers.

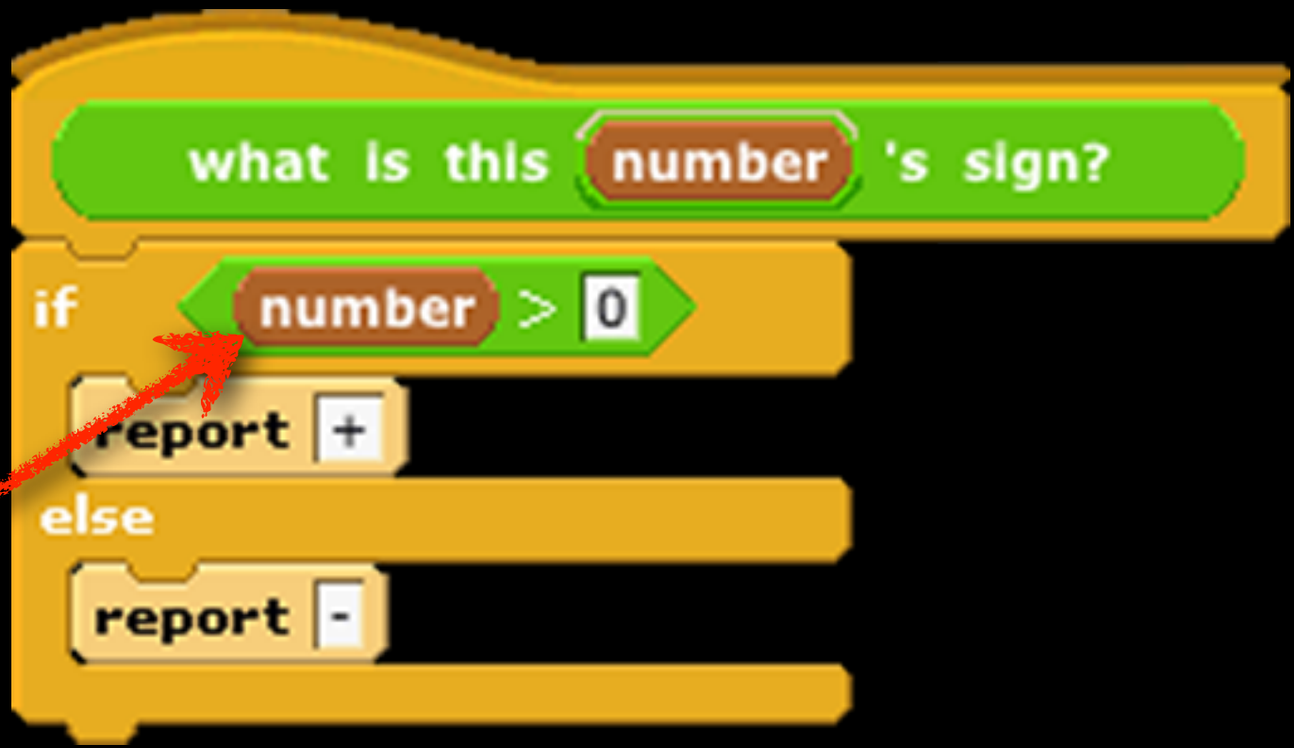
No! Why?



Conditionals

Reports “+” for positive numbers, “-” for negative numbers.

What about number = 0?

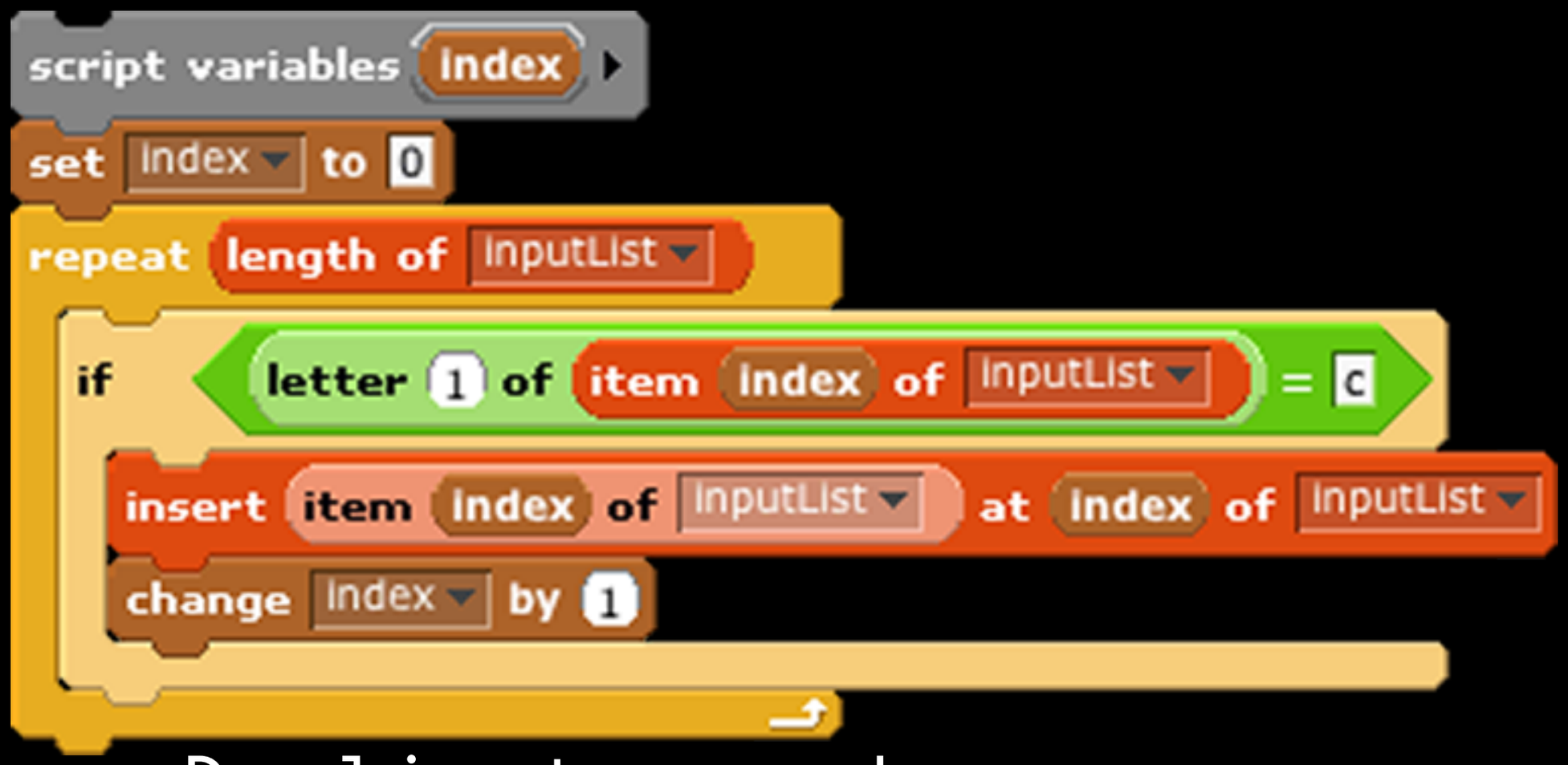


Conditionals

Loops & Variables

- Be sure that your conditionals handle ***all*** possible cases!
- Double-check *edge cases*, or inputs that fall on either side of your predicate.

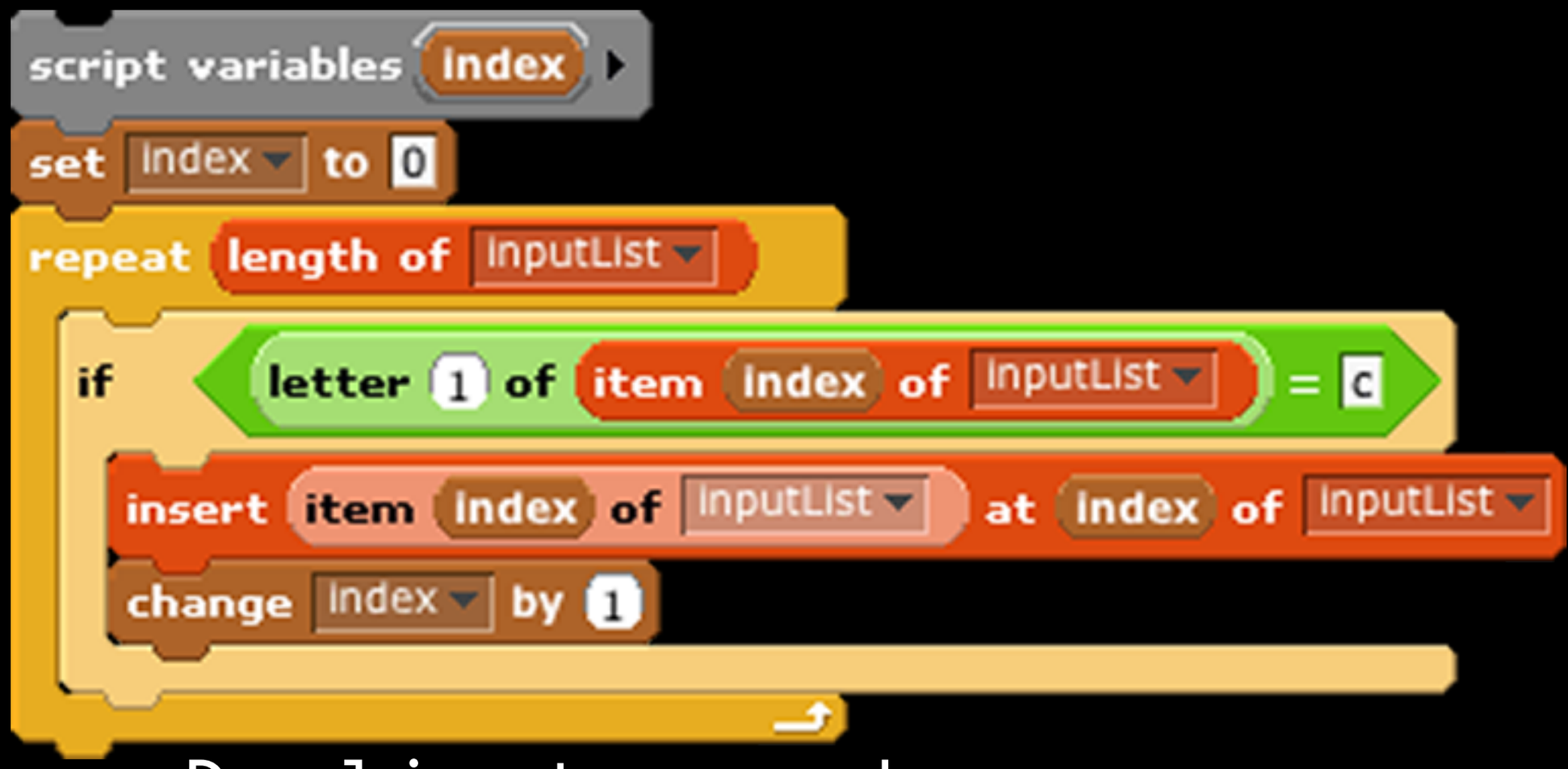
Lists



Duplicate words beginning with ‘c’

Does this correctly loop over list?

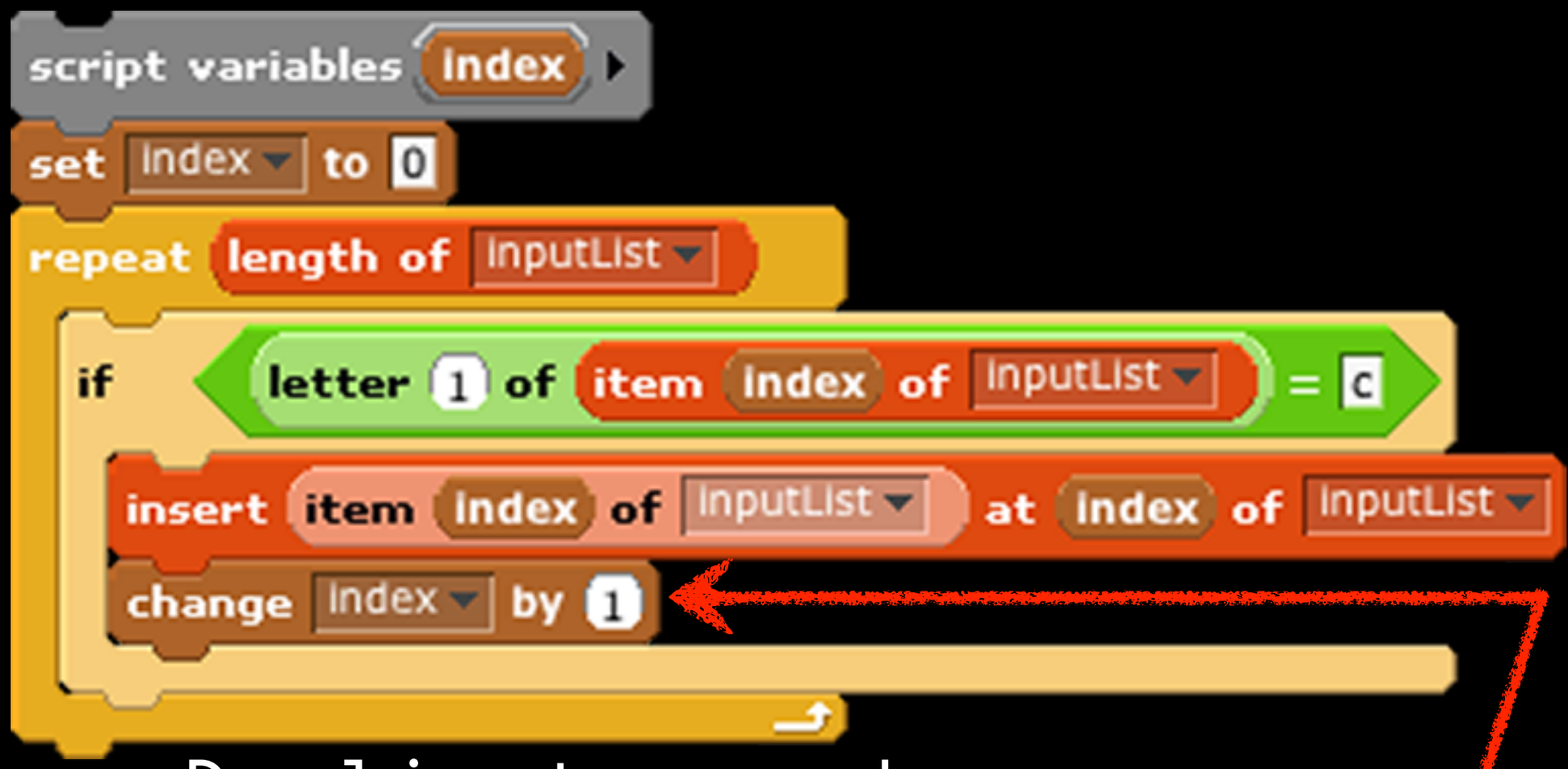
Lists



Duplicate words beginning with ‘c’

No! Why?

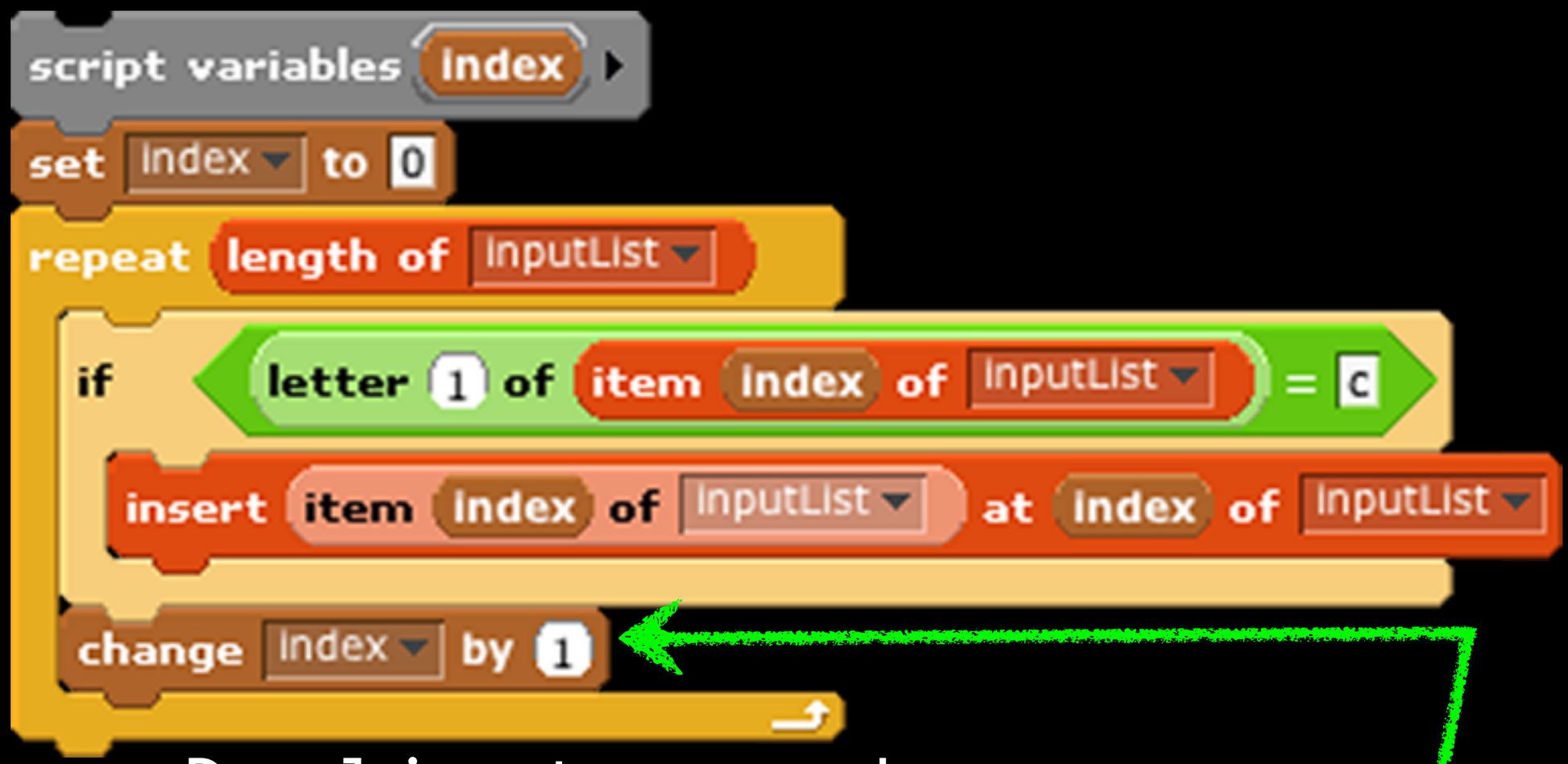
Lists



Duplicate words beginning with ‘c’

Index within conditional!

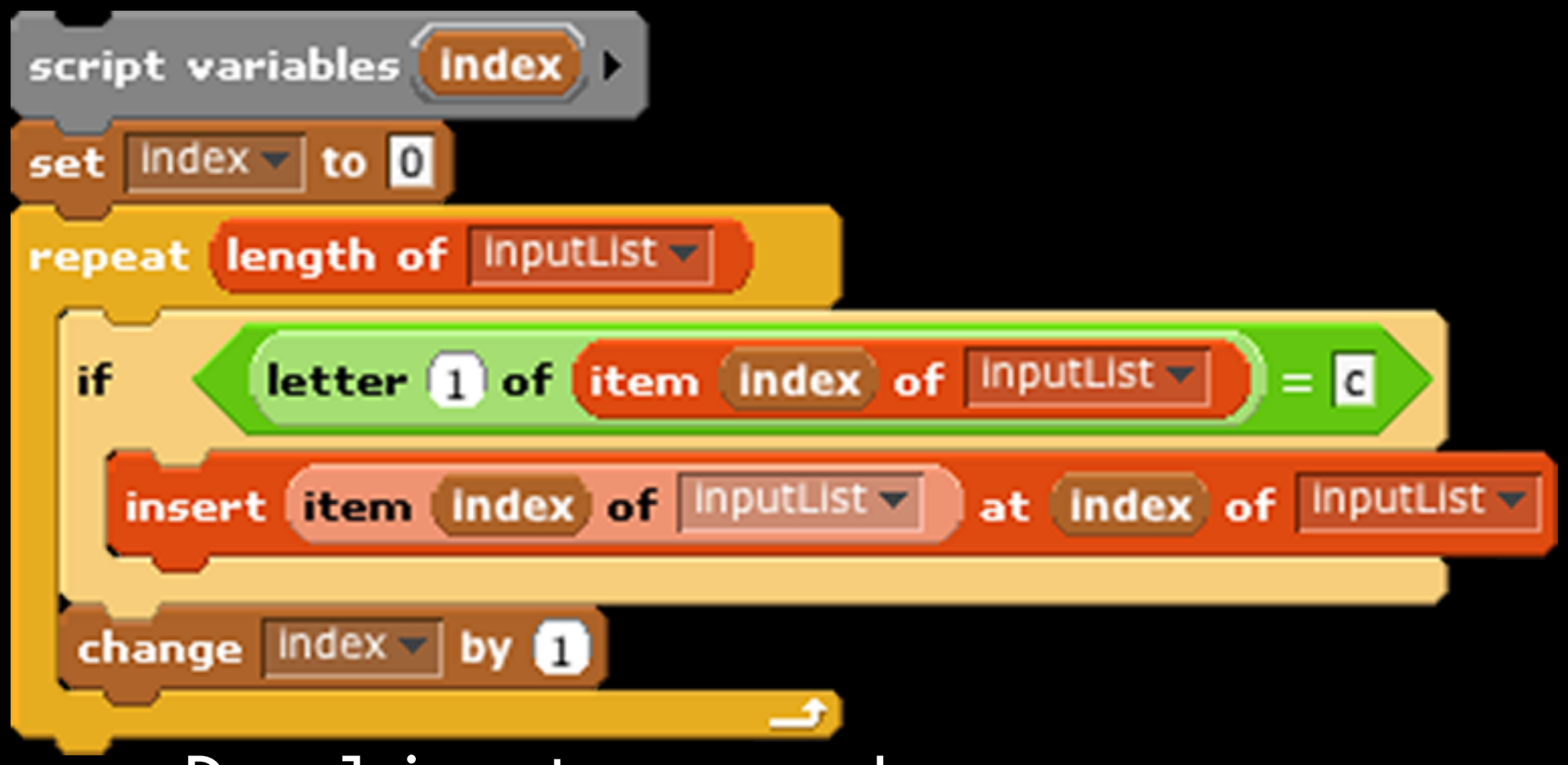
Lists



Duplicate words beginning with ‘c’

Correct now?

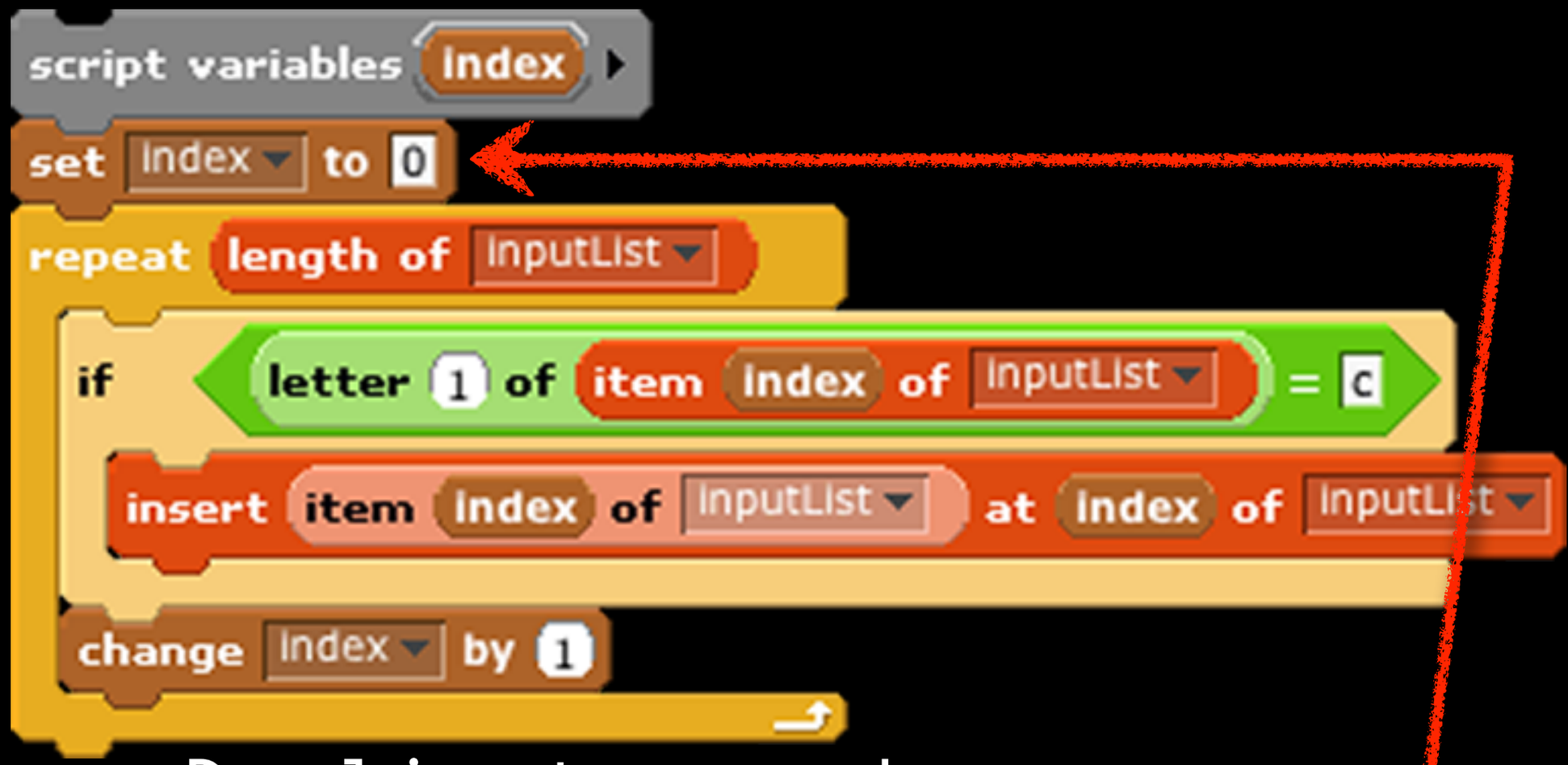
Lists



Duplicate words beginning with ‘c’

No! Why?

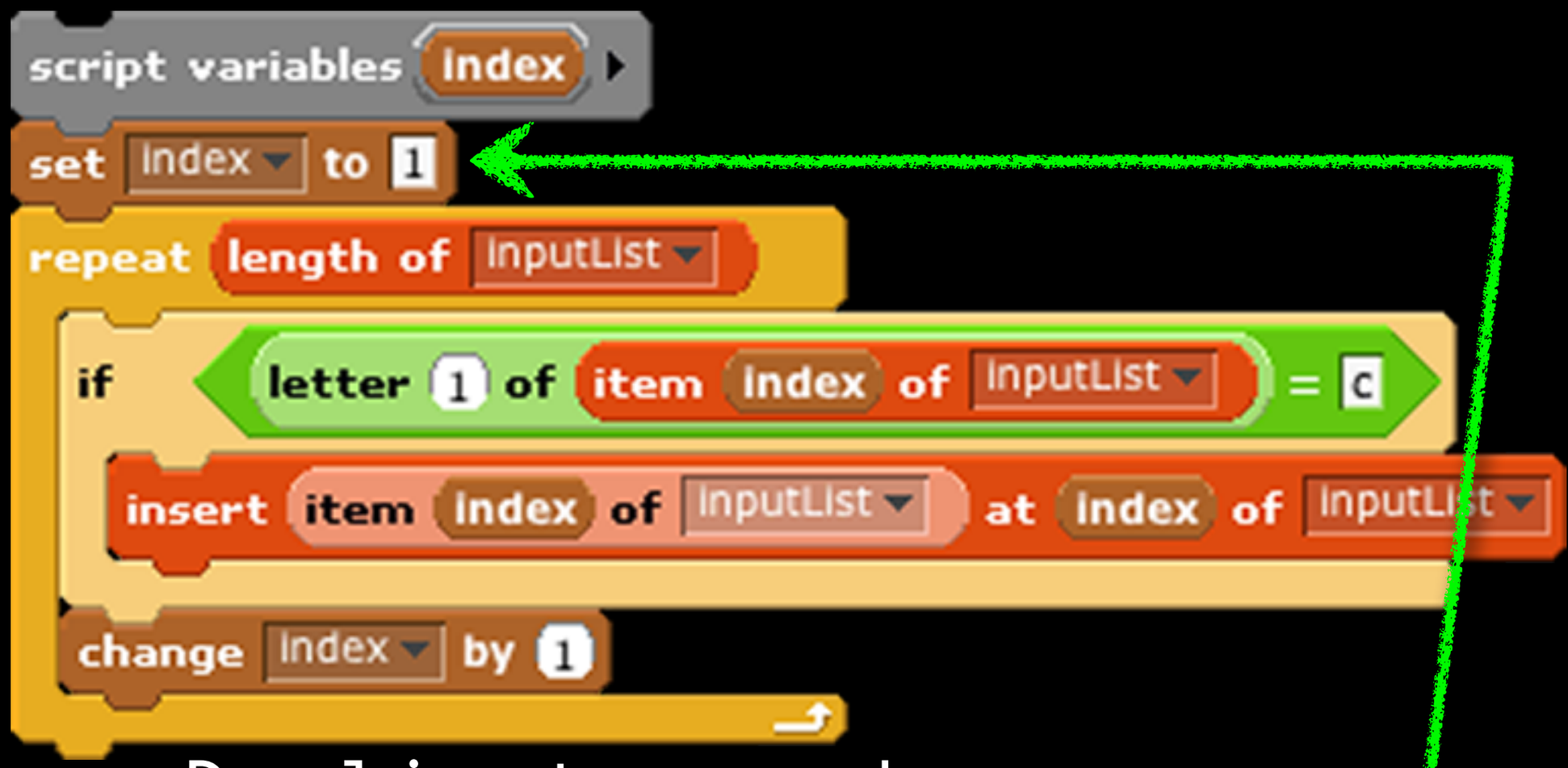
Lists



Duplicate words beginning with ‘c’

Off by 1!

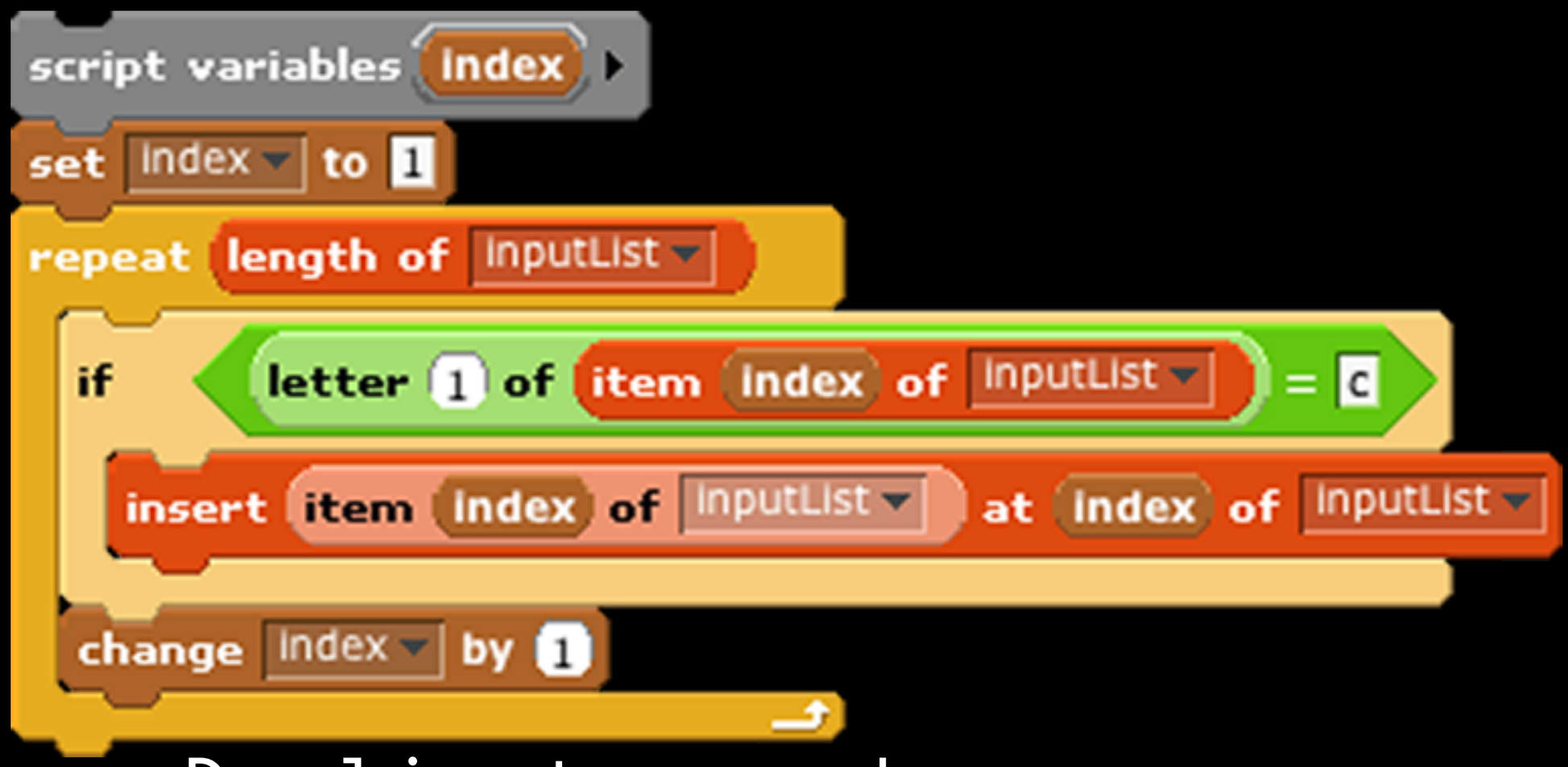
Lists



Duplicate words beginning with ‘c’

Correct now?

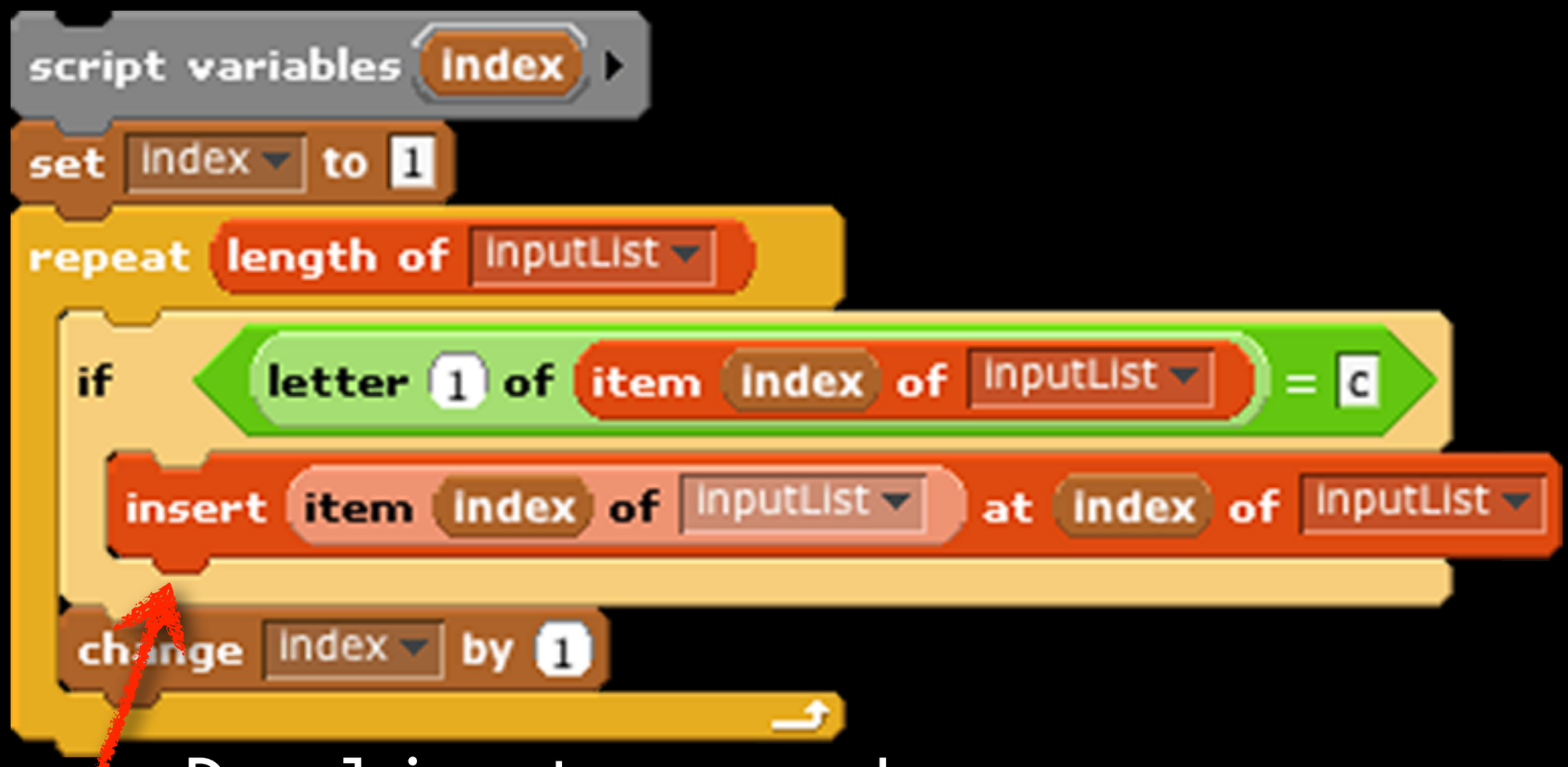
Lists



Duplicate words beginning with ‘c’

No! Why?

Lists



Duplicate words beginning with ‘c’

The list keeps changing size!

Lists

Duplicate words beginning with ‘c’

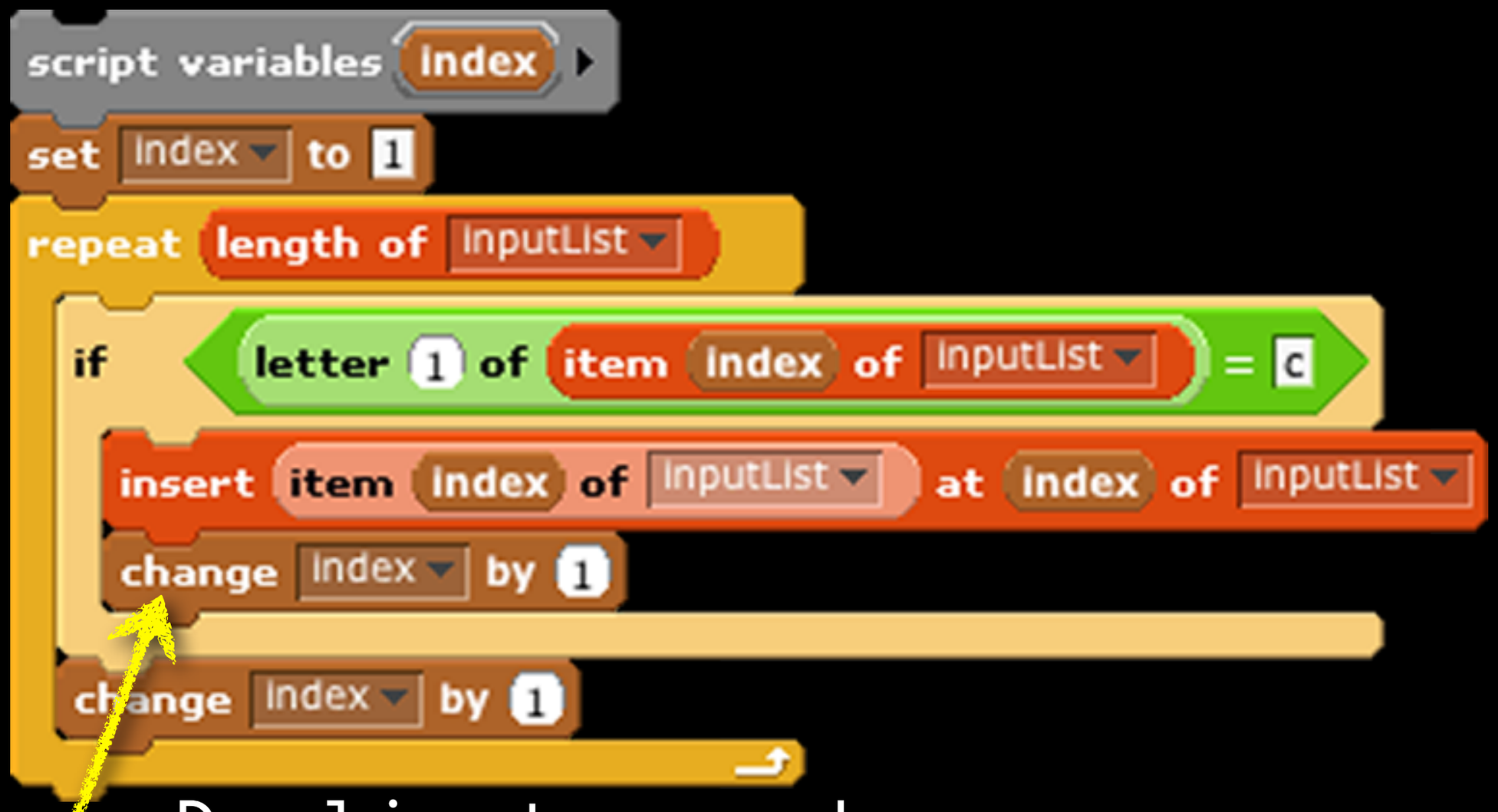
The list keeps changing size!

Lists

How do you correct it?

Here is one solution...

Lists



Duplicate words beginning with ‘c’

Push the index past the inserted item...

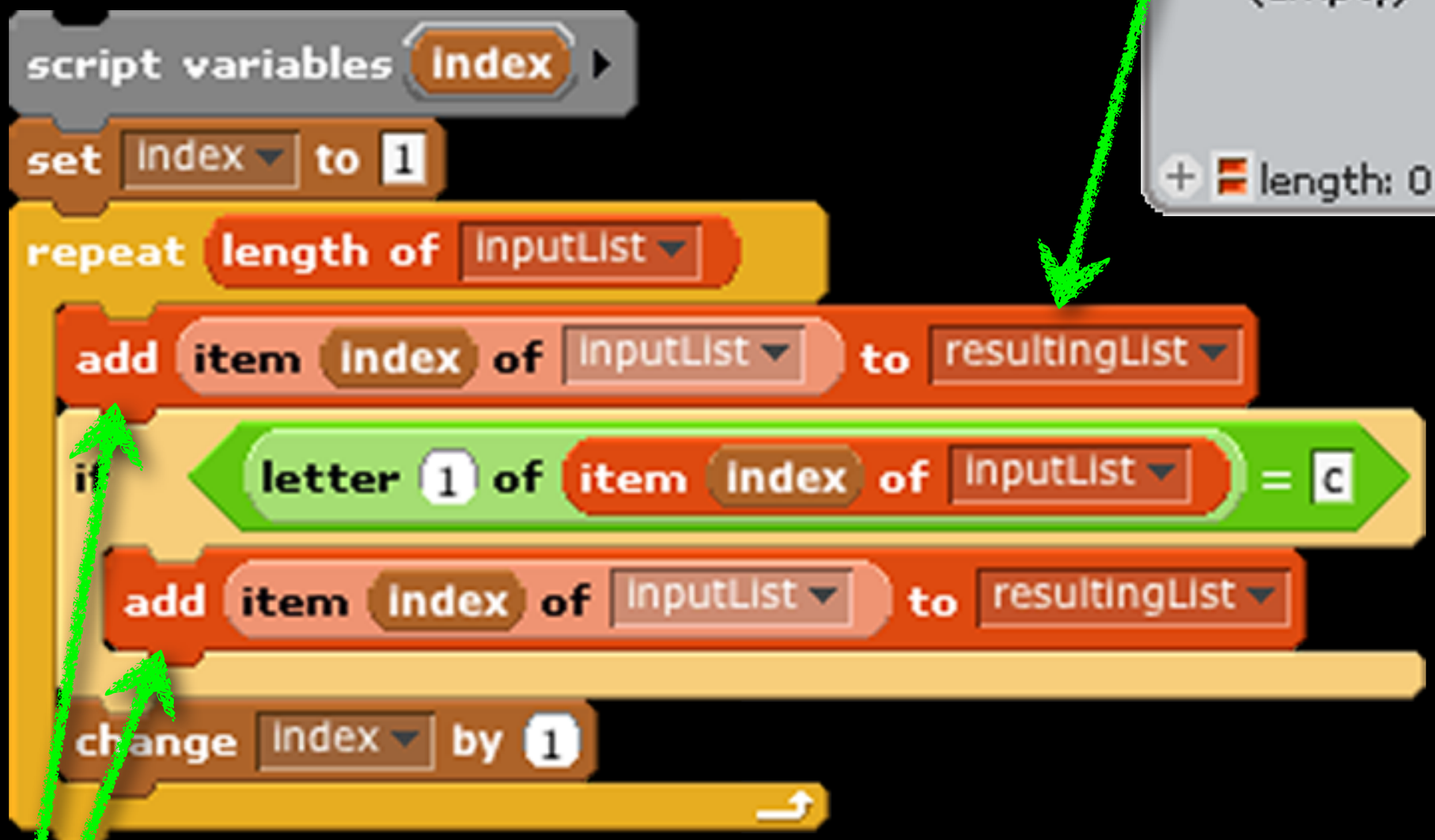
Lists

Seems pretty “kludgy” though...

Here is a much, much better solution...

http://en.wikipedia.org/wiki/Kludge

Lists

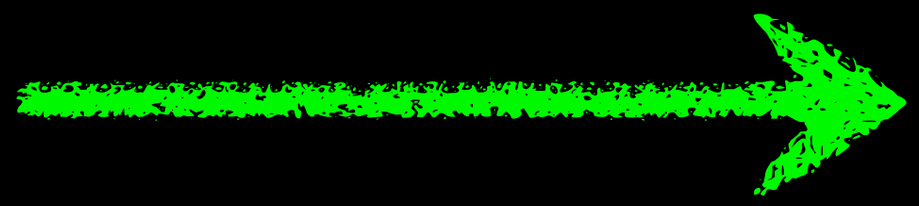


Duplicate words beginning with ‘c’

Make a new, resulting list!

Lists

Duplicate words beginning with ‘c’



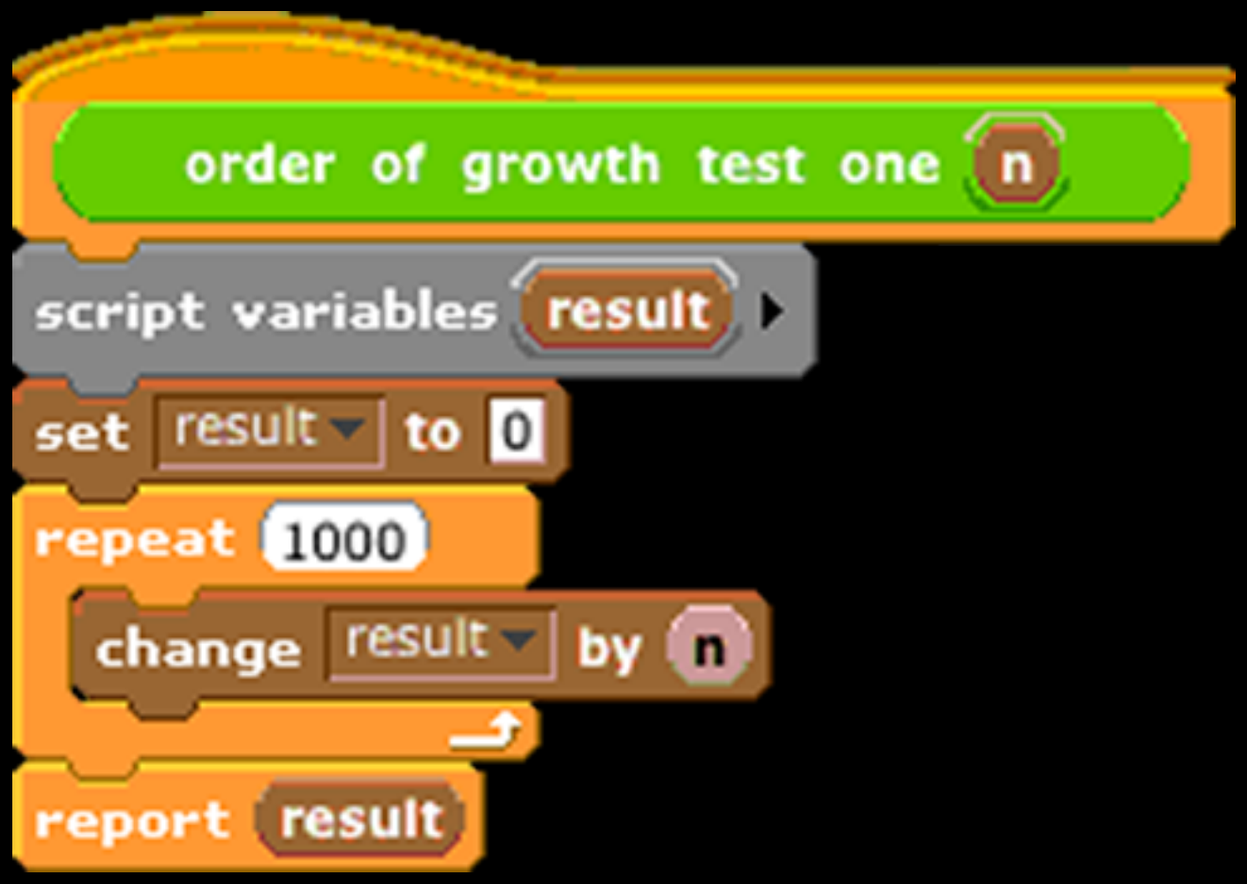
Much better! And our original list is still intact!

Check-Lists

- Be sure indexes are correct during the *entire* loop.

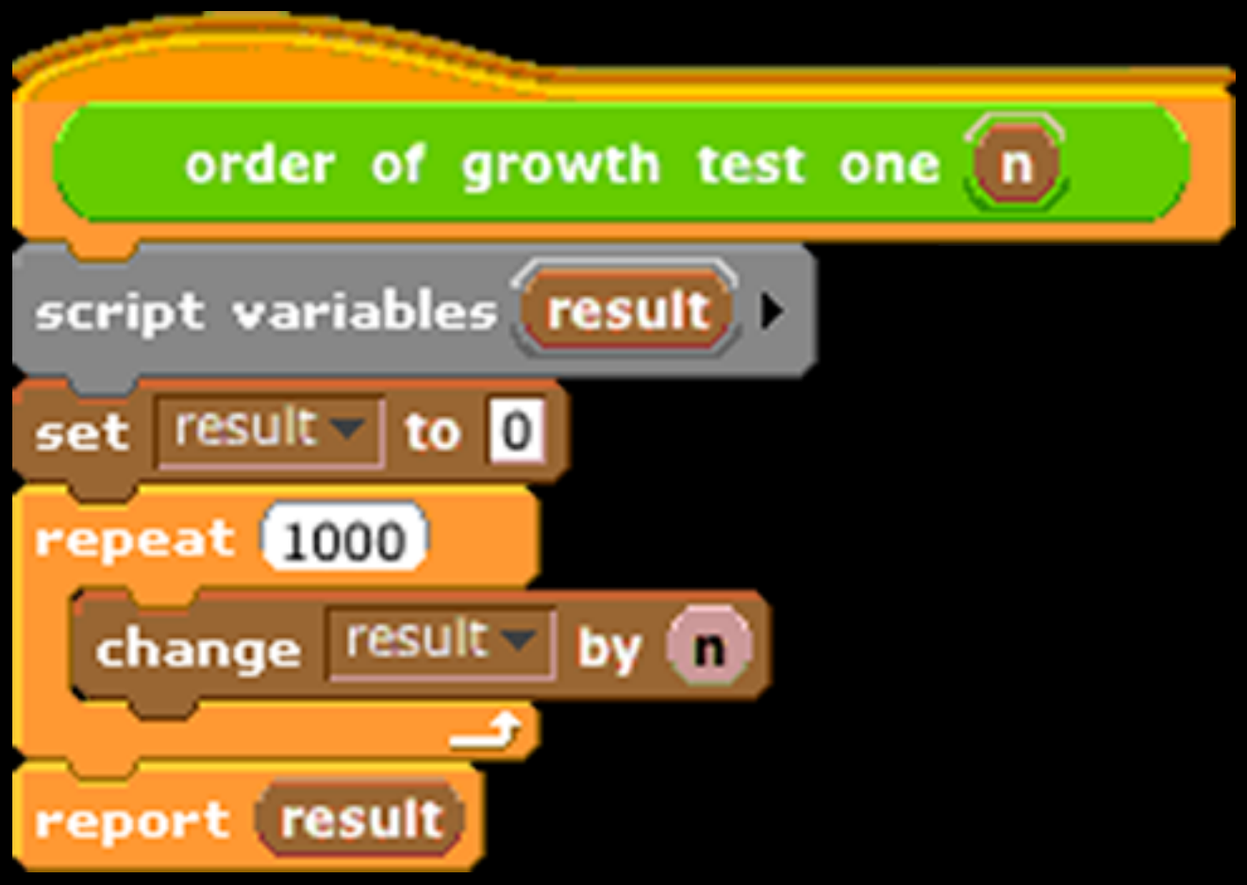
- BYOB starts list indexing at 1
- If the input list changes size during the loop, your index will be off!

Algorithms & Complexity



Order of growth?

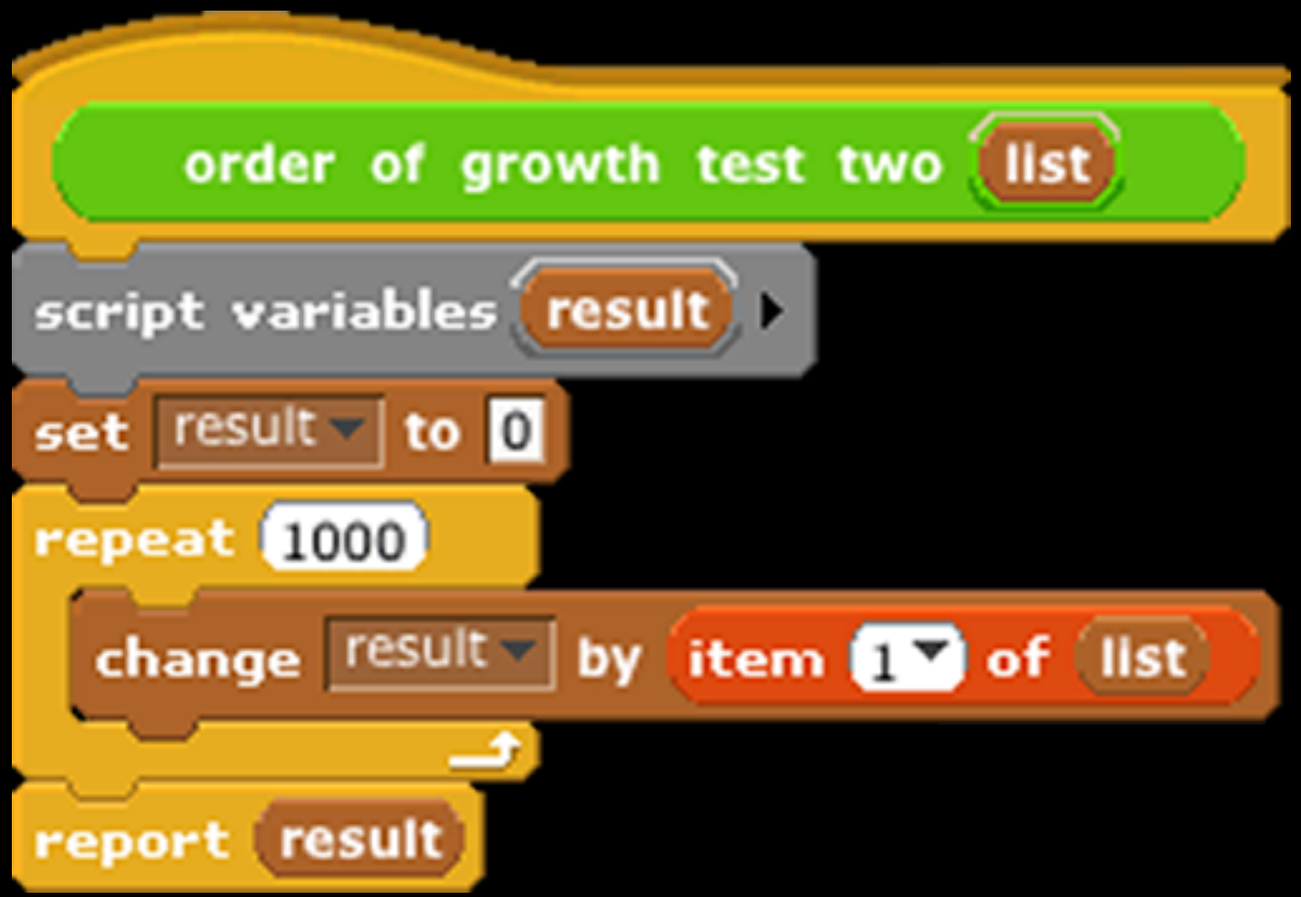
Algorithms & Complexity



Constant: O(c)

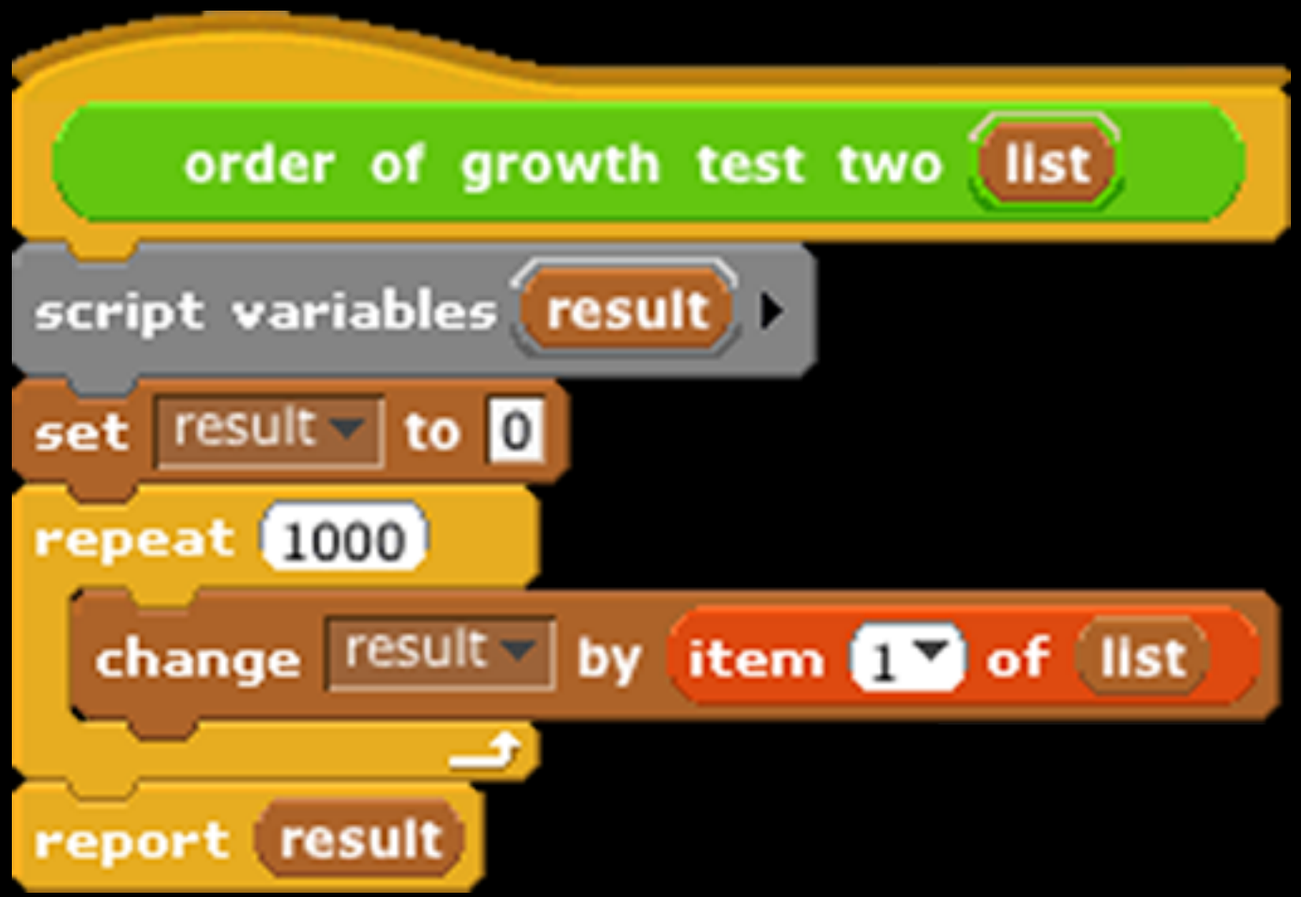
Reason: No matter what "n" is, the loop *always* repeats 1000 times.

Algorithms & Complexity



Order of growth?

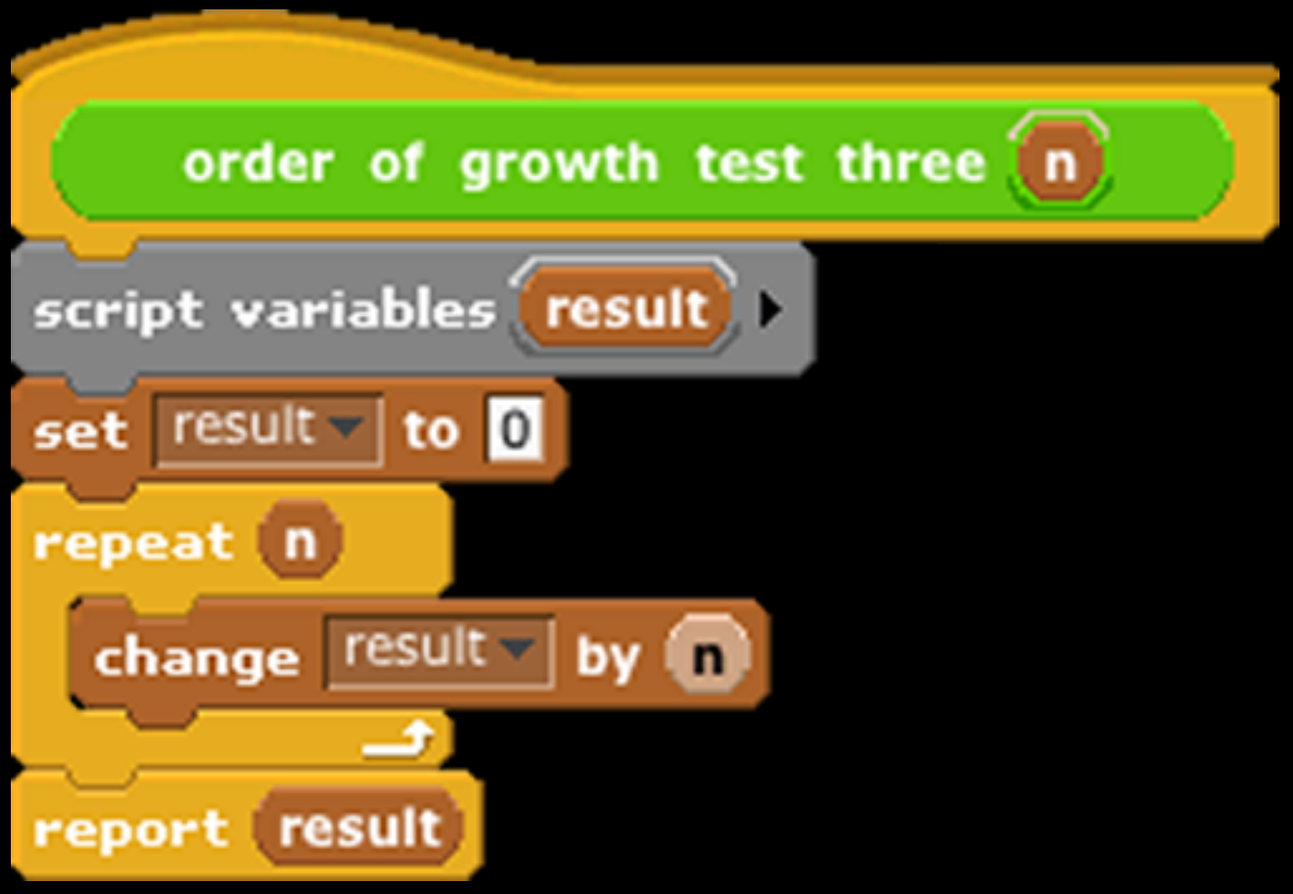
Algorithms & Complexity



Still Constant: O(c)

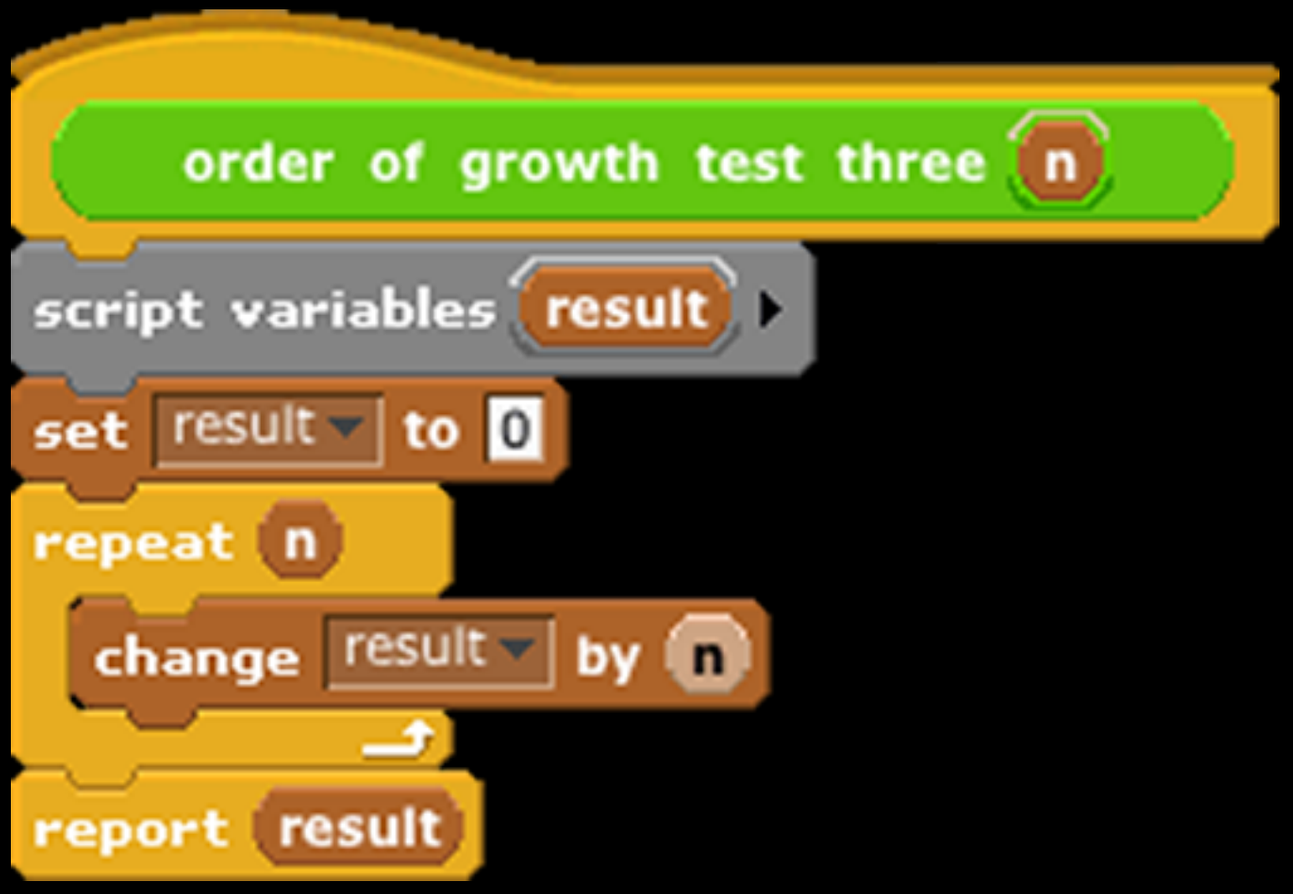
Reason: No matter what the list is, the loop *always* repeats 1000 times.

Algorithms & Complexity



Order of growth?

Algorithms & Complexity



Linear: O(n)

Reason: Number of operations proportional to the input size (n)

Algorithms & Complexity

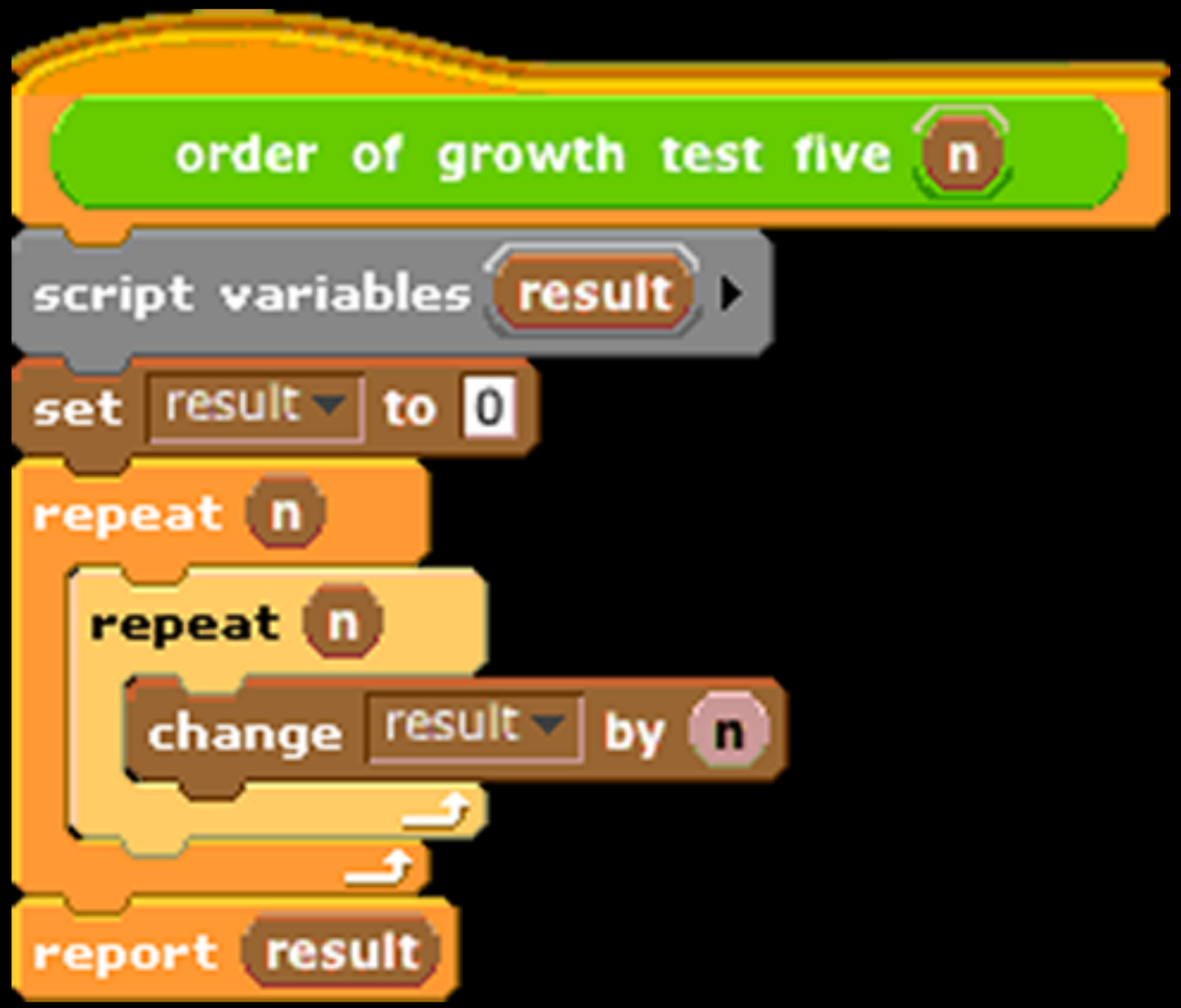
Order of growth?

Algorithms & Complexity

Still Linear: O(n)

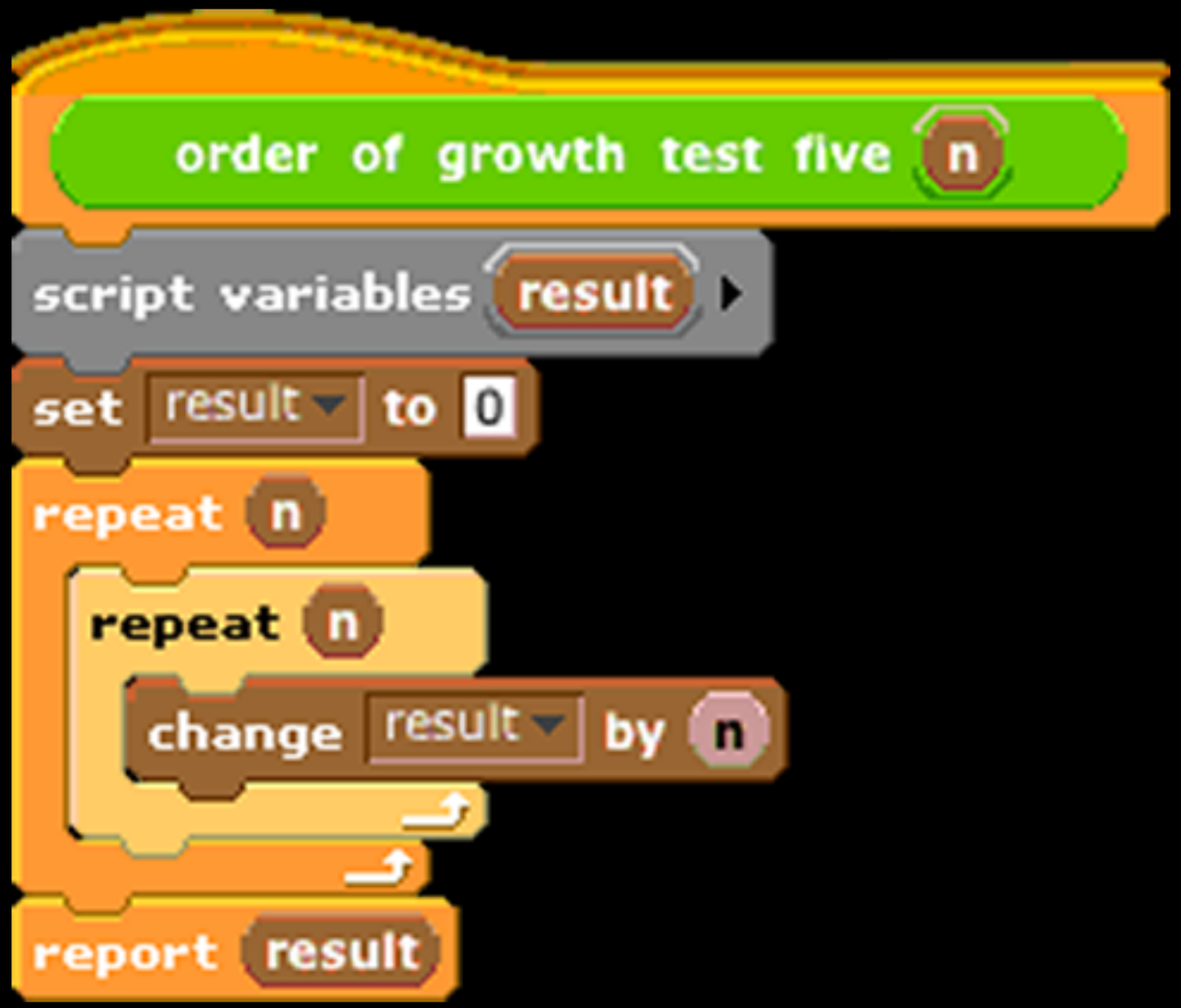
Reason: Number of operations proportional to the input size (length)

Algorithms & Complexity



Order of growth?

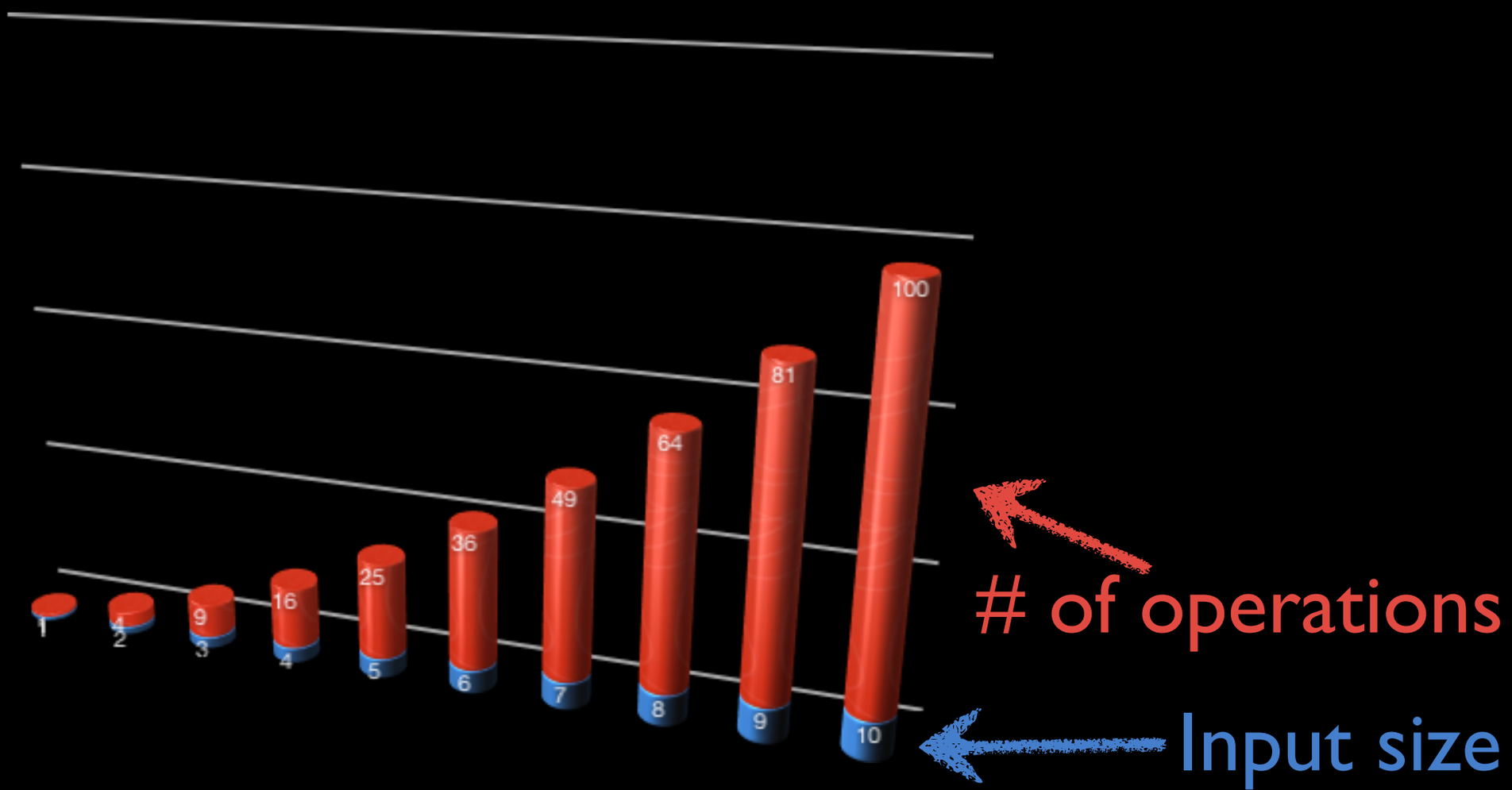
Algorithms & Complexity



Quadratic: $O(n^2)$

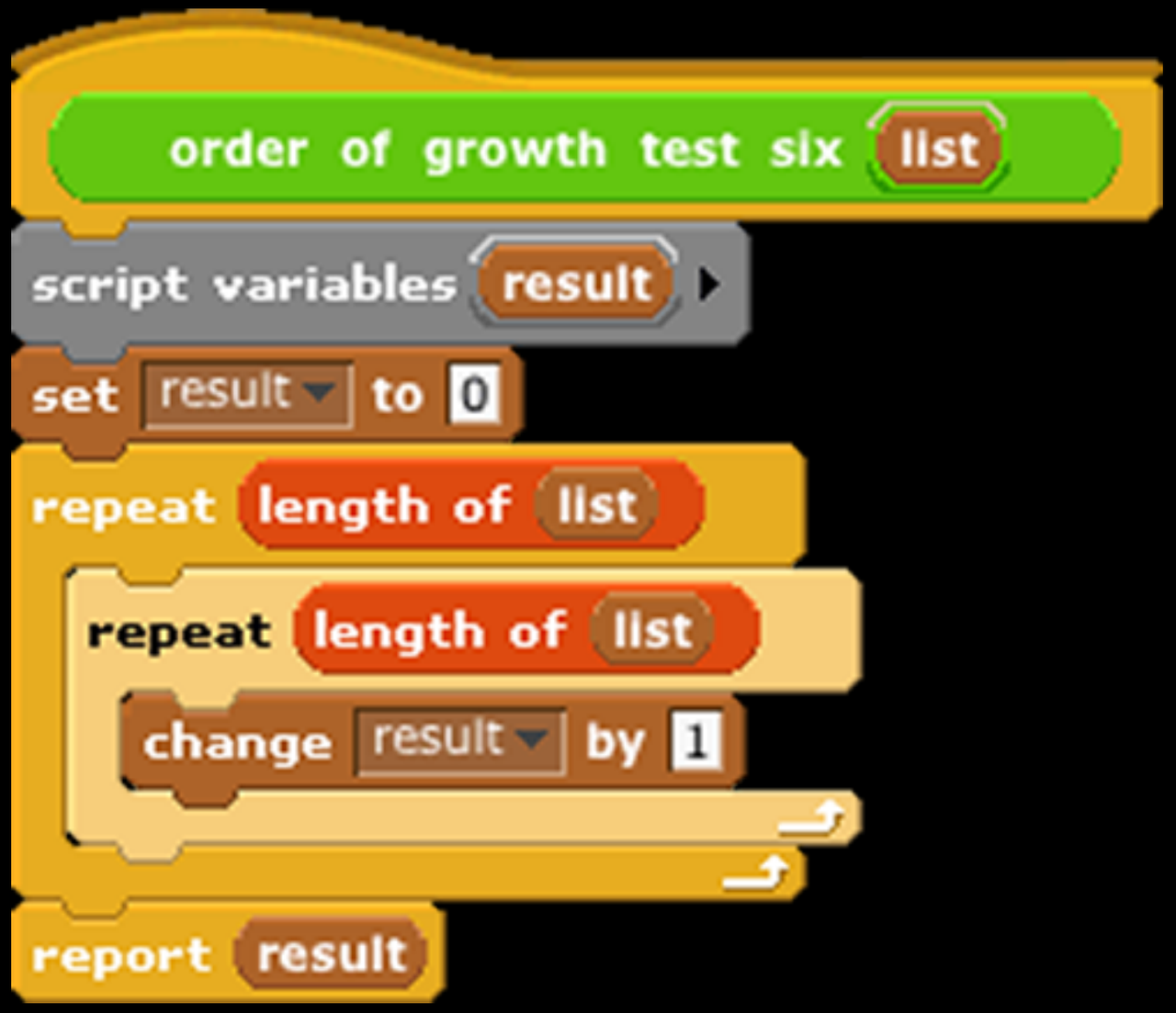
Reason: Number of operations proportional to the square of the size of the input data (n)

Algorithms & Complexity



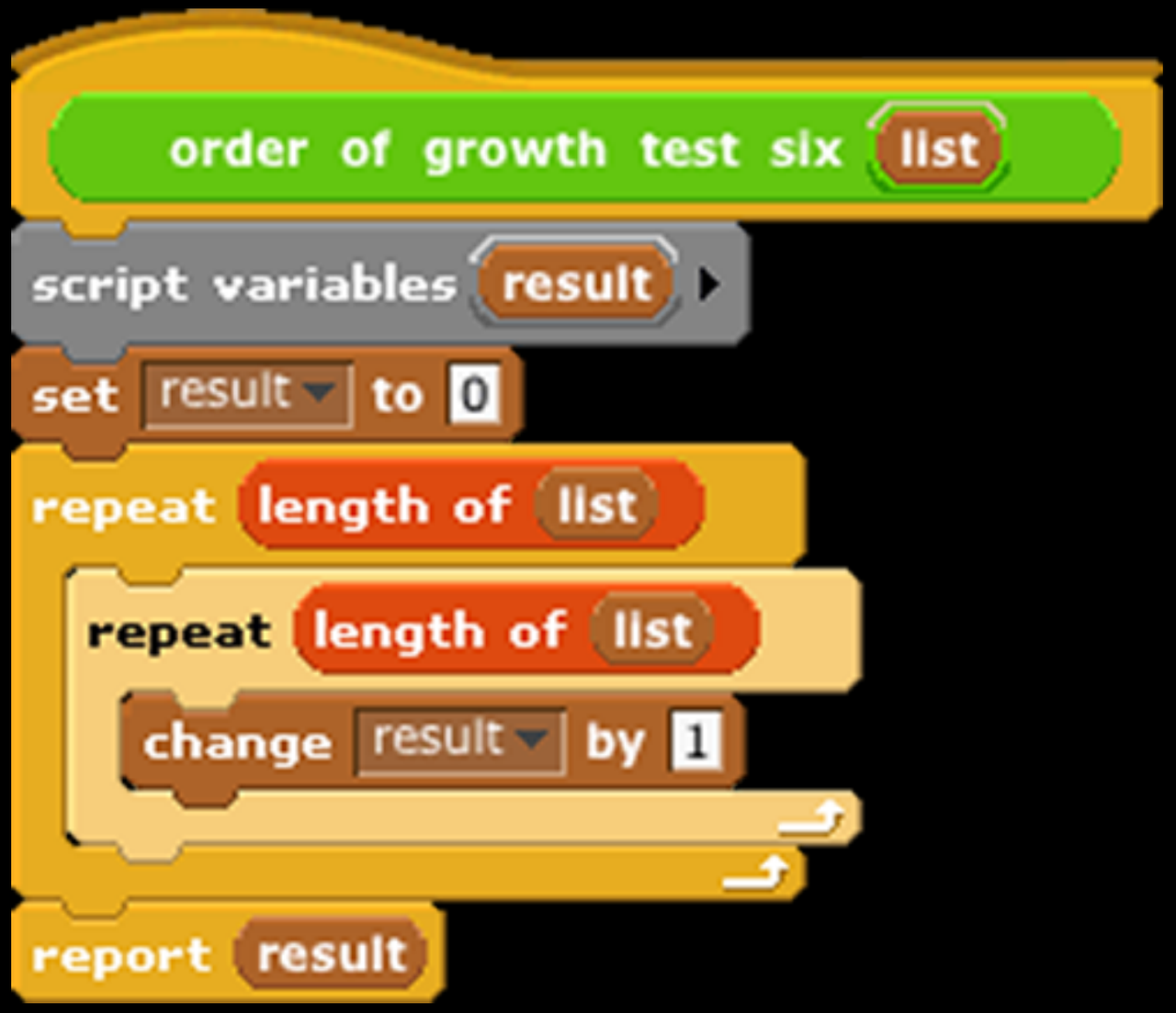
Quadratic: $O(n^2)$

Algorithms & Complexity



Order of growth?

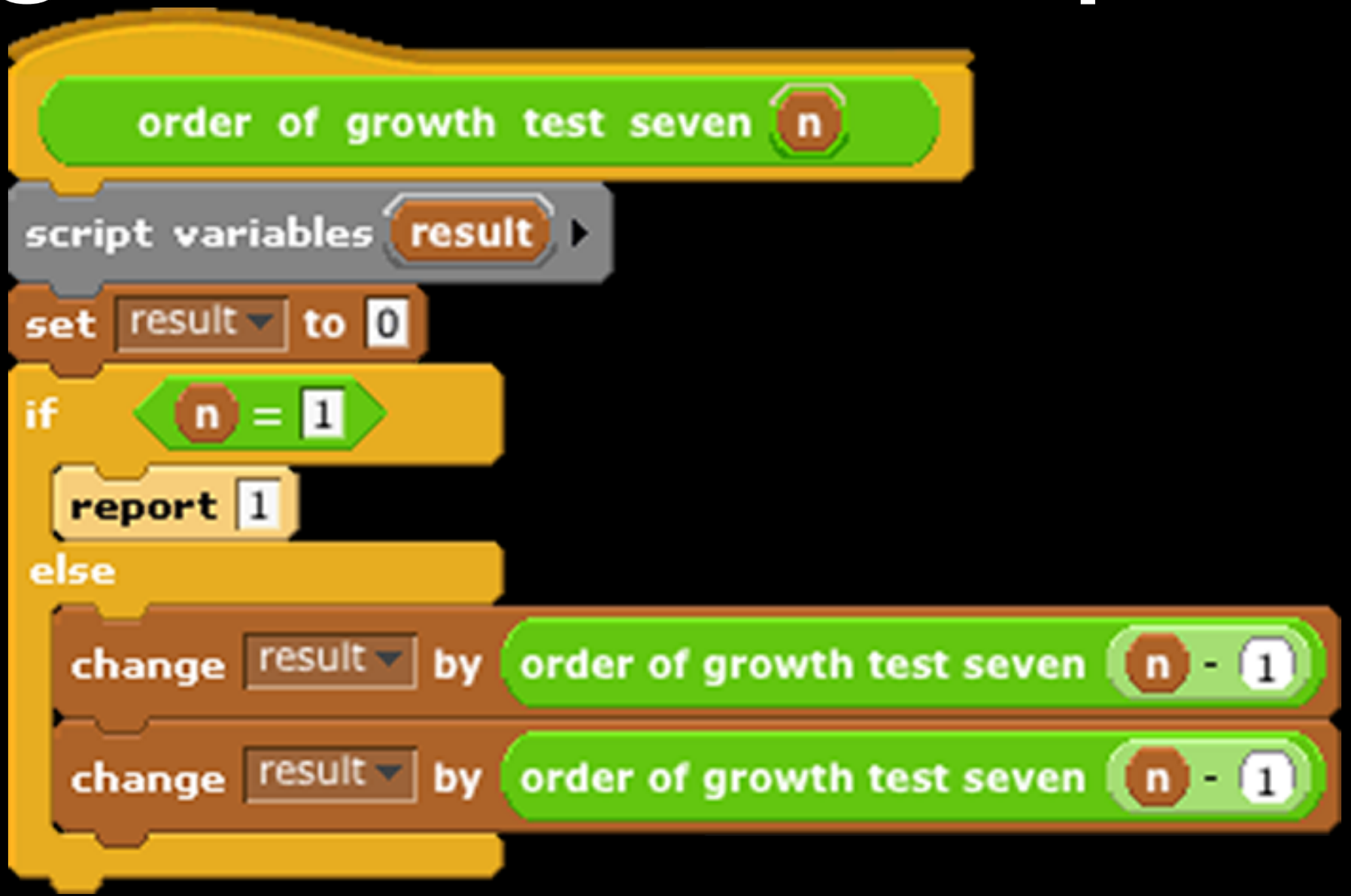
Algorithms & Complexity



Still Quadratic: $O(n^2)$

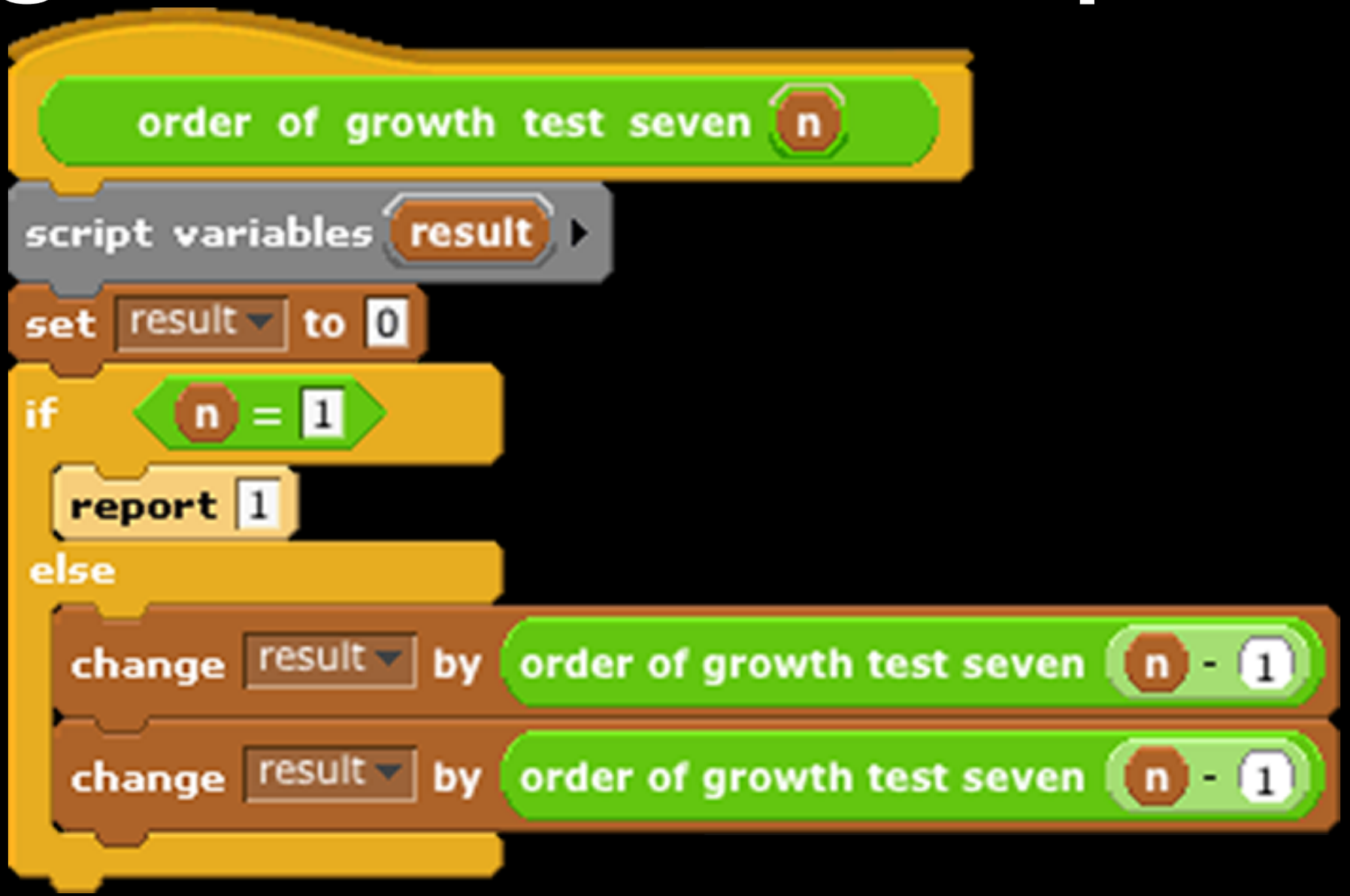
Reason: Number of operations proportional to the square of the size of the input data (length)

Algorithms & Complexity



Order of growth?

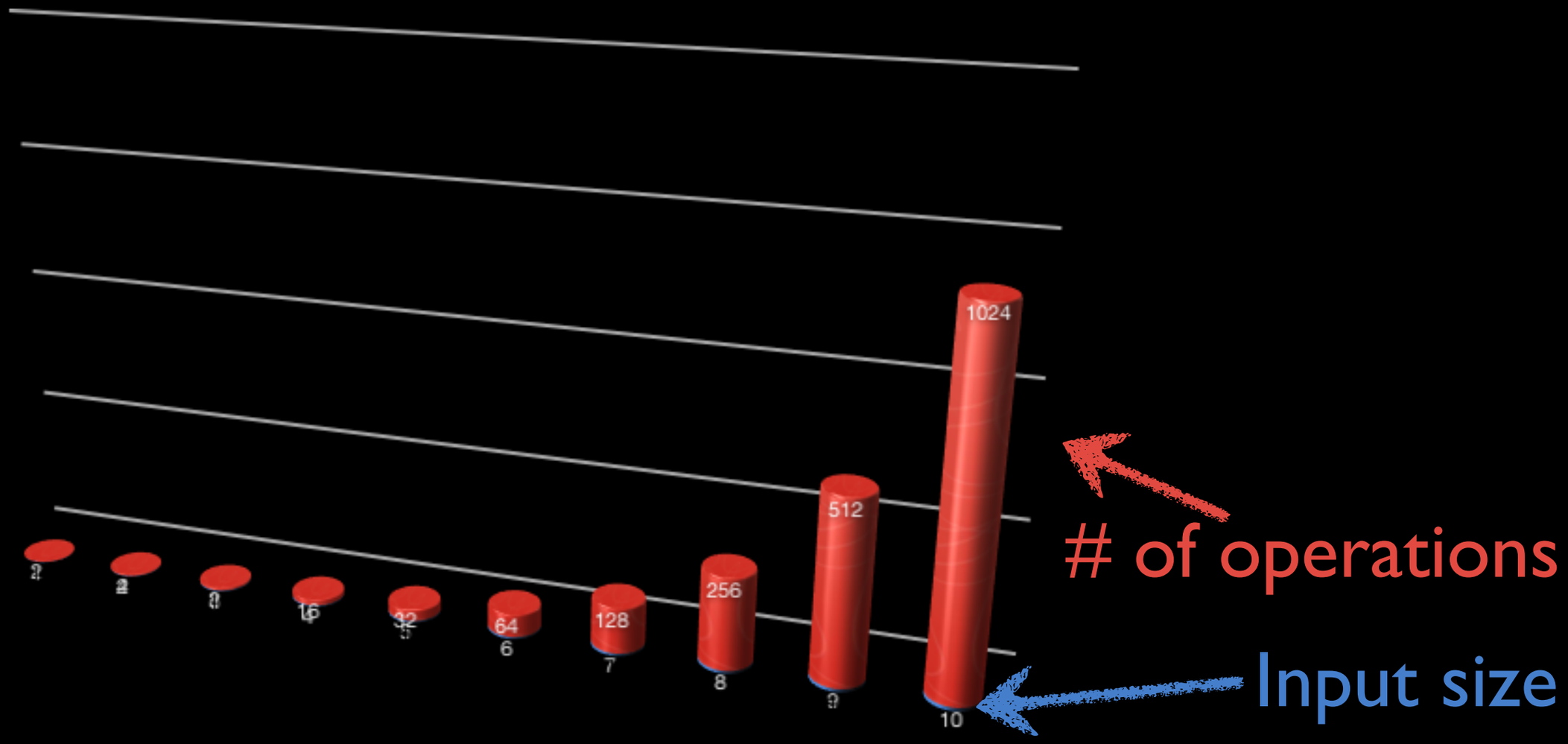
Algorithms & Complexity



Exponential: $O(c^n)$

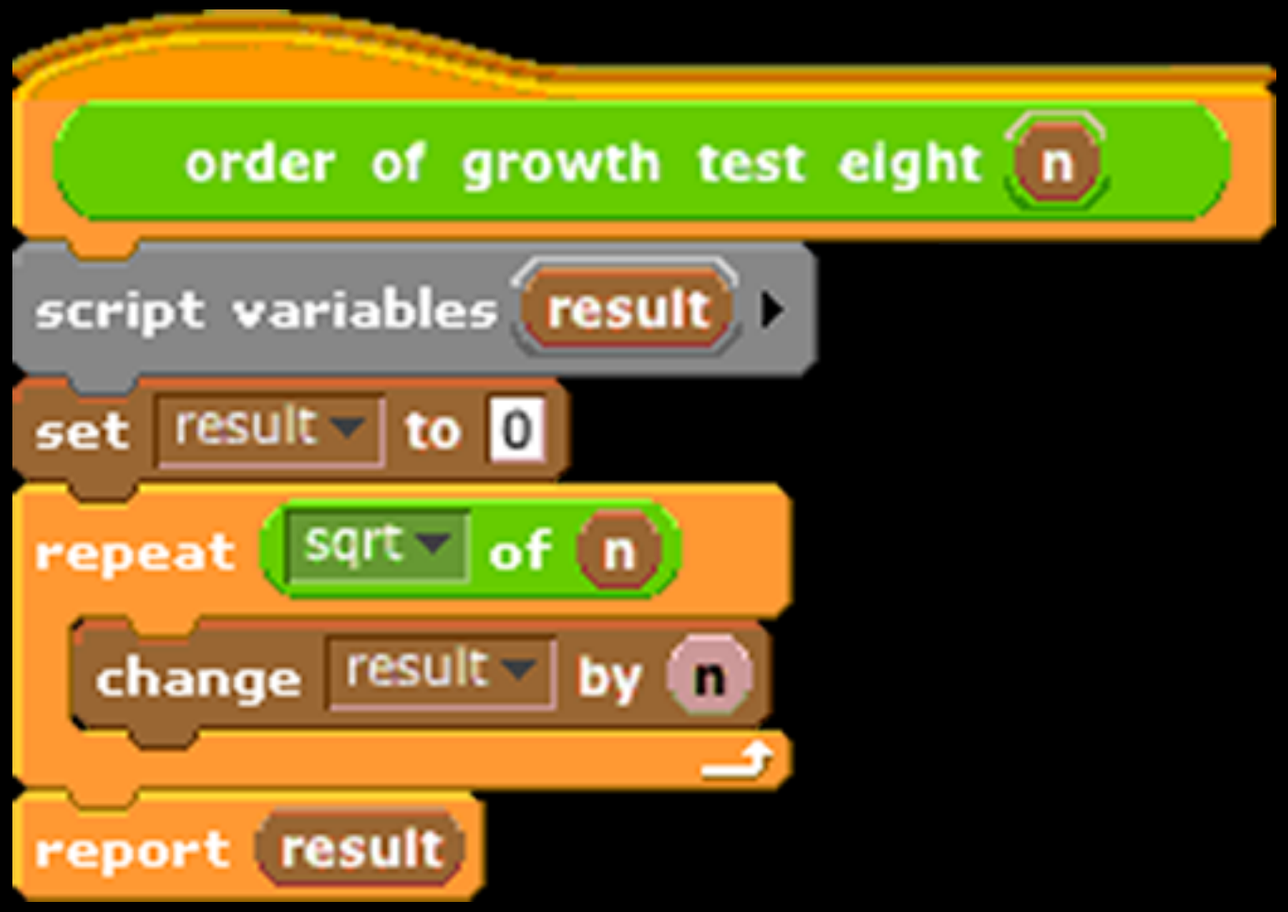
Reason: the recursive call is run twice for each value of n.

Algorithms & Complexity



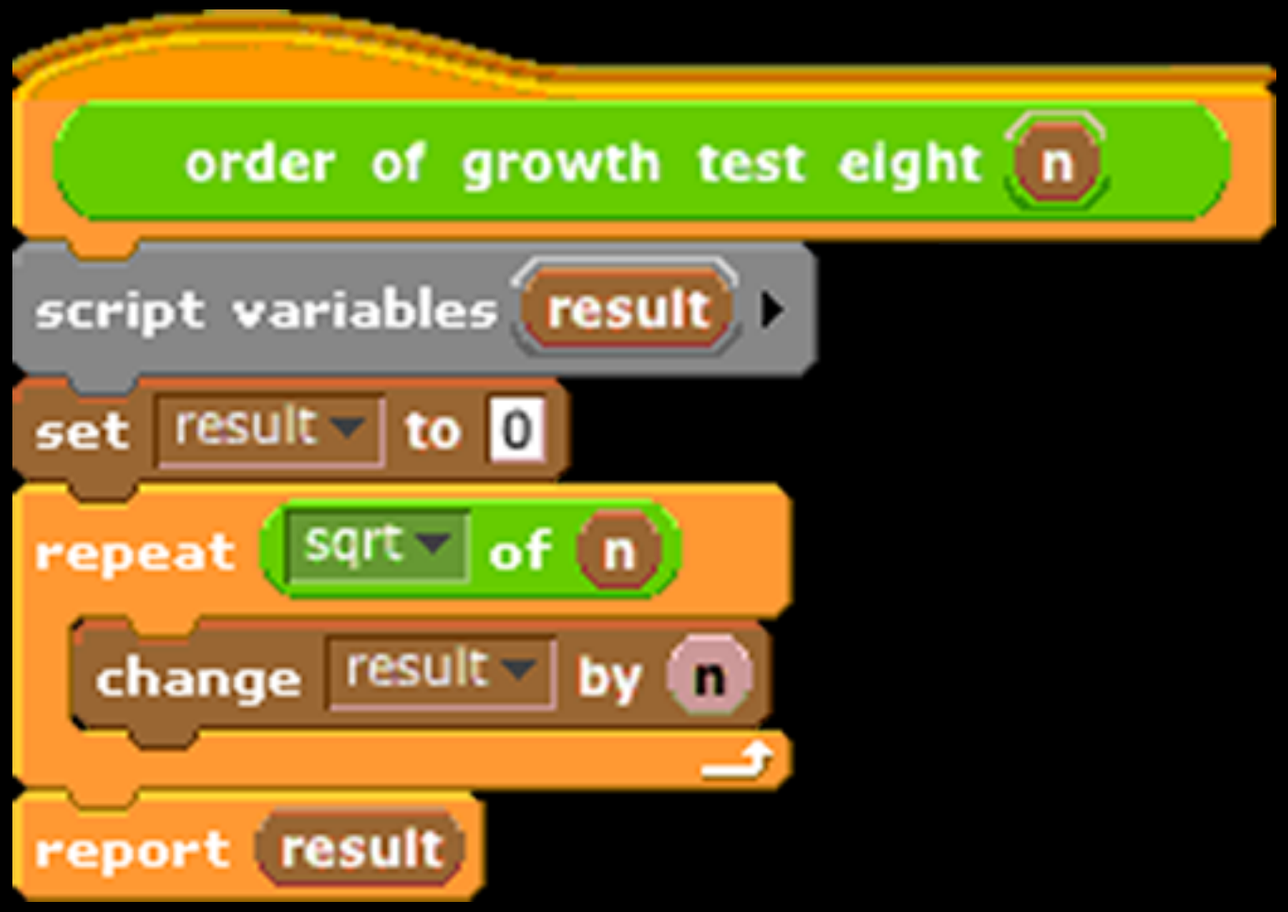
Exponential: $O(2^n)$

Algorithms & Complexity



Order of growth?

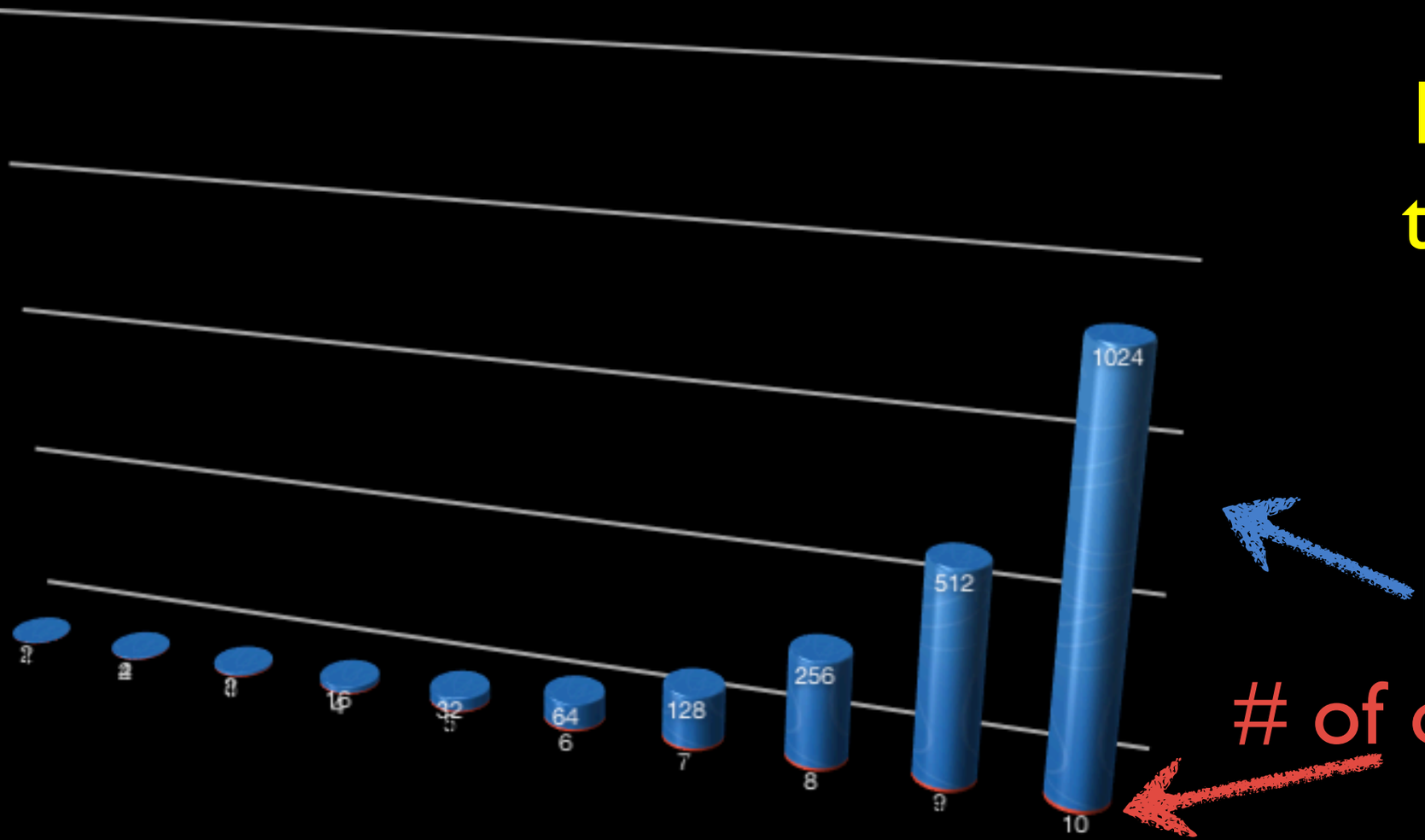
Algorithms & Complexity



Logarithmic: $O(Log_c n)$

Reason: the number of times the loop runs grows far more slowly (square root) than "n"

Algorithms & Complexity



Note that these have traded places!

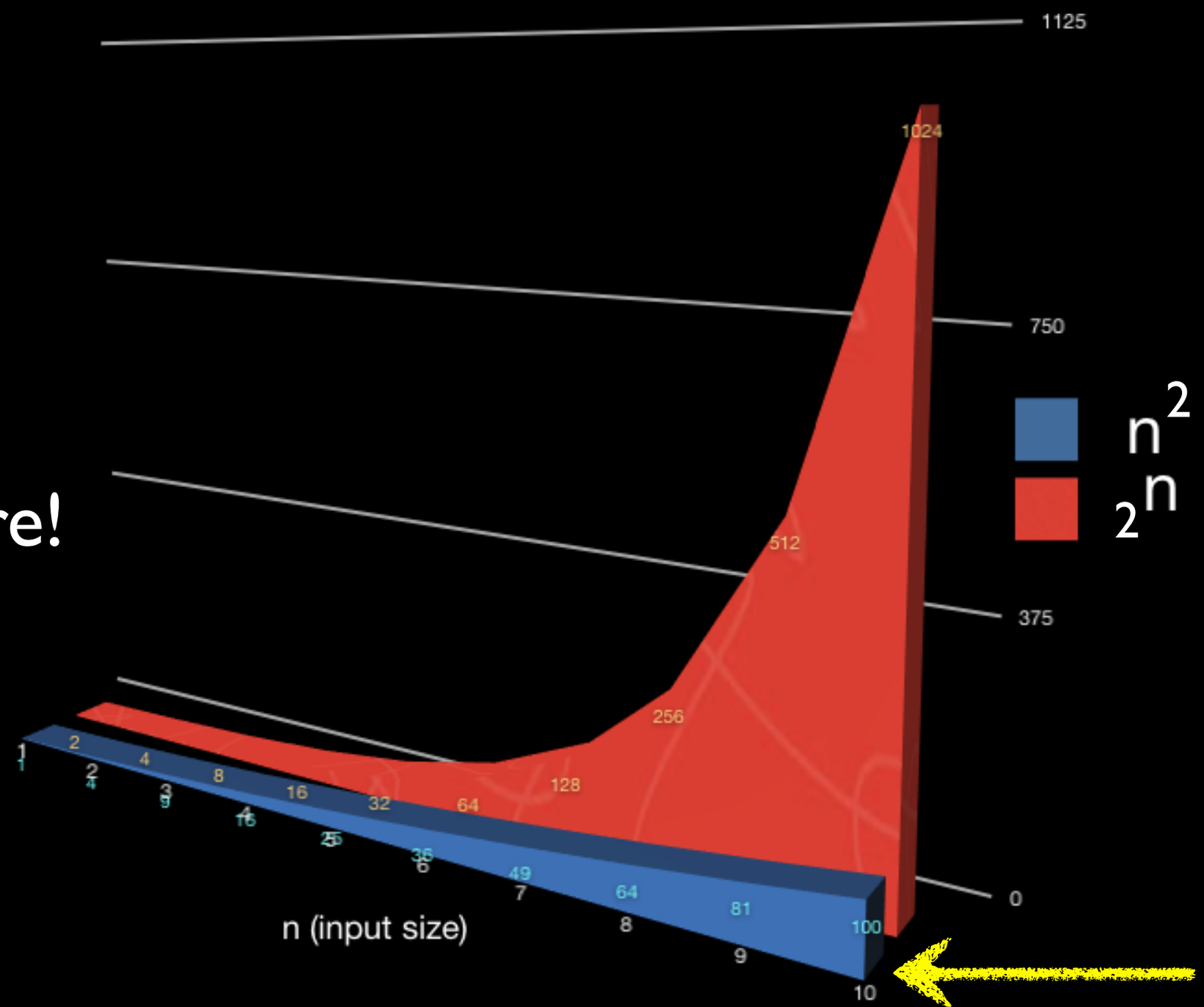
Input size

# of operations

Logarithmic: $O(Log_2 n)$

Algorithms & Complexity

Compare!



Note that linear growth doesn't even register on this graph!

Algorithms & Complexity

Consider the number of times the calculation *repeats*, rather than specific inputs.

Algorithms & Complexity Examples

Constant → Hash Table Lookup

Linear → Text or List Length

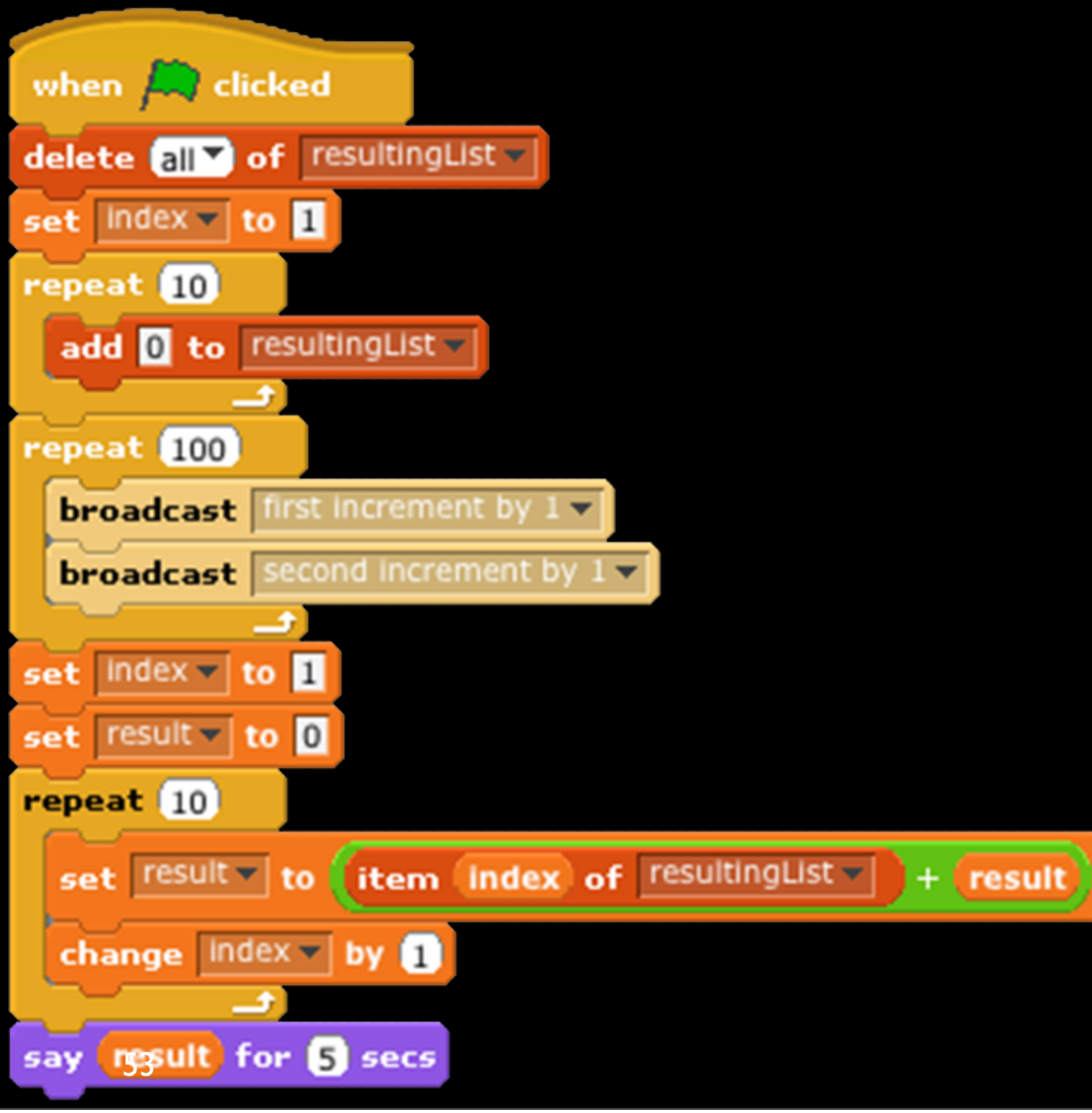
Quadratic → Square Matrix Calcs.
→ (Naïve) Sorting

Exponential → Recursive Fractals

Logarithmic → Binary Searches

Concurrency

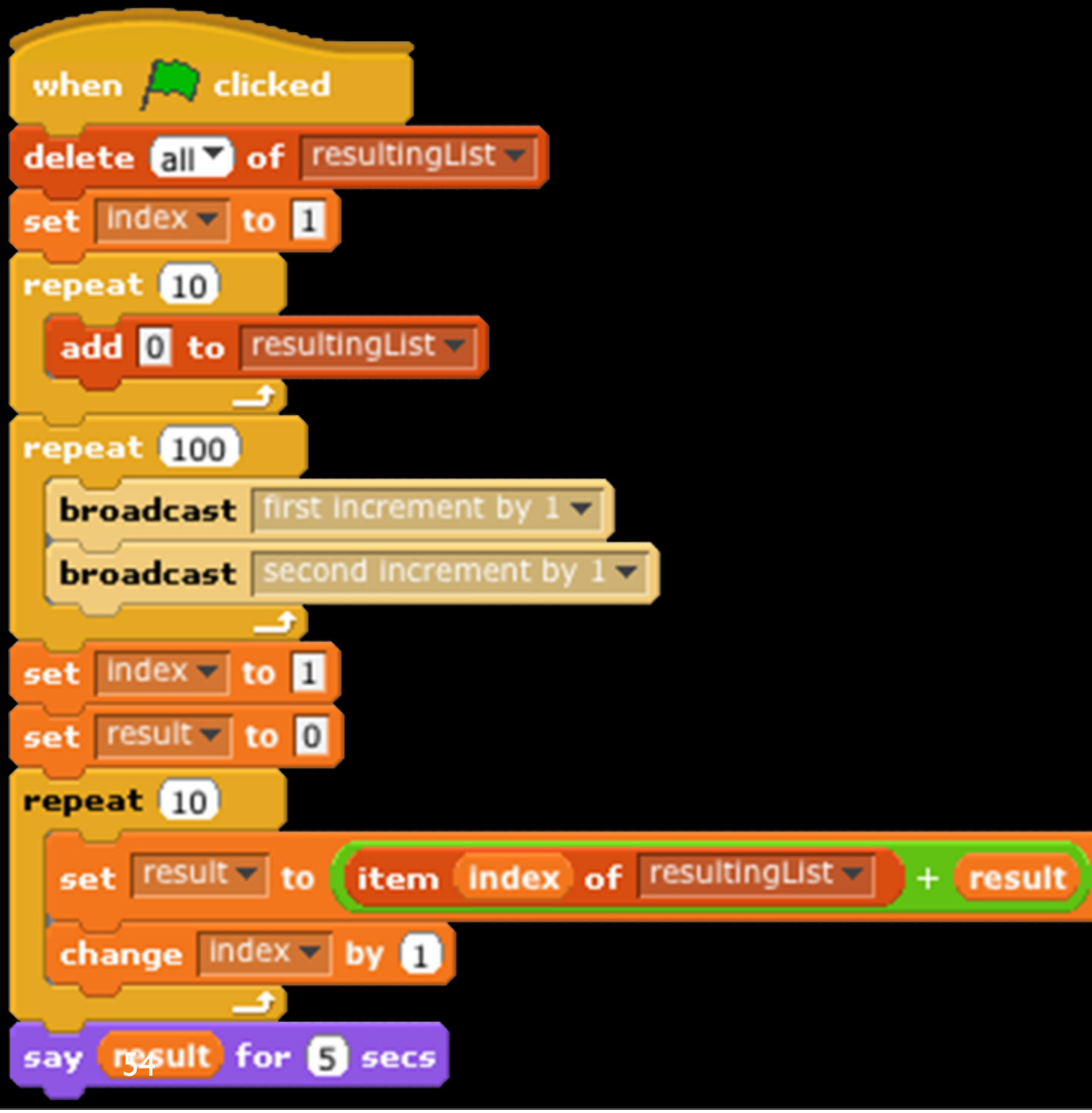
List adds up to 200?



Concurrency

List adds up to 200?

No! Why?



Concurrency

These might choose the same number at the same time!

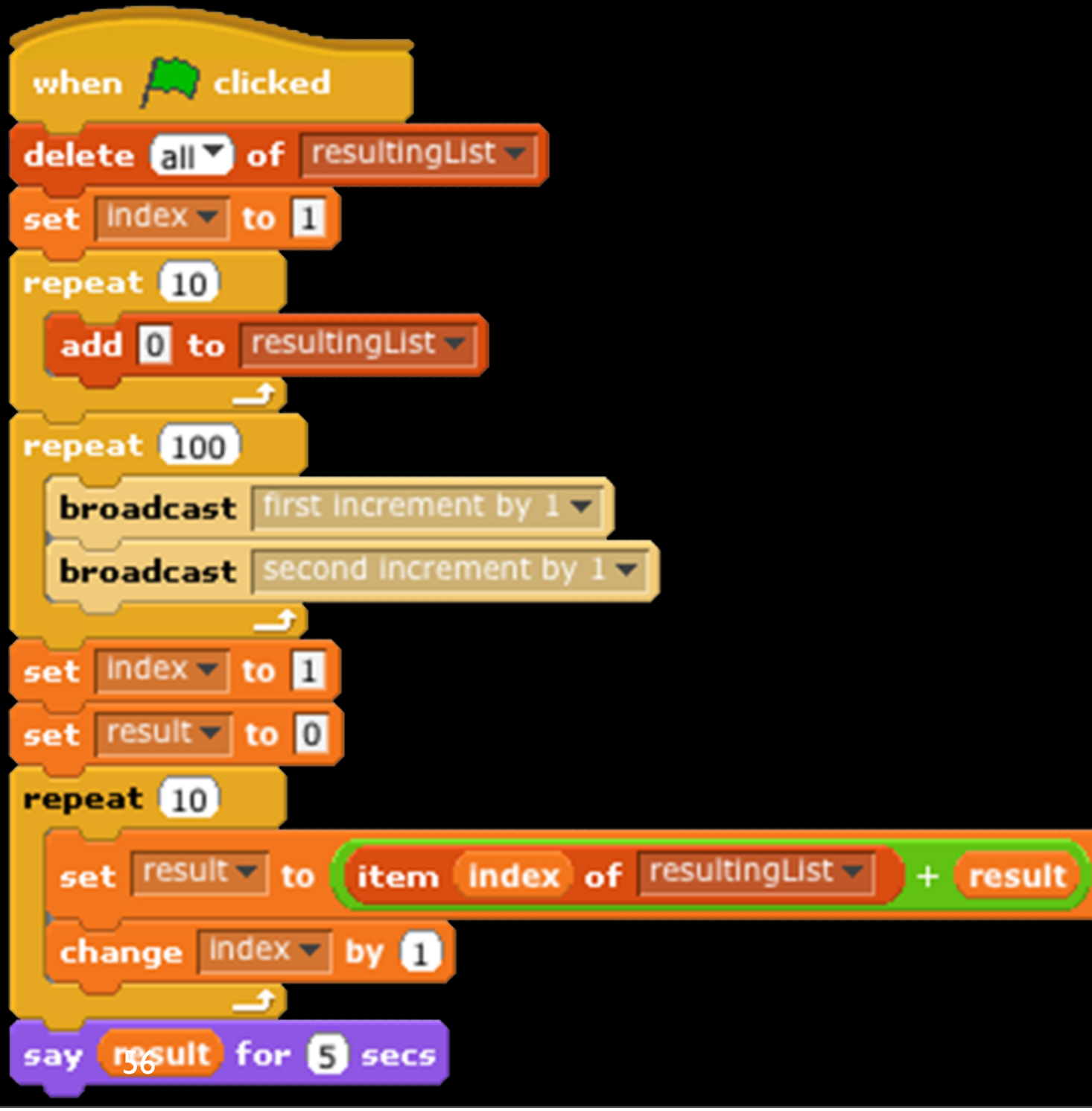
List adds up to 200?

No! Why?

Concurrency

List adds up to 200?

Anything else?



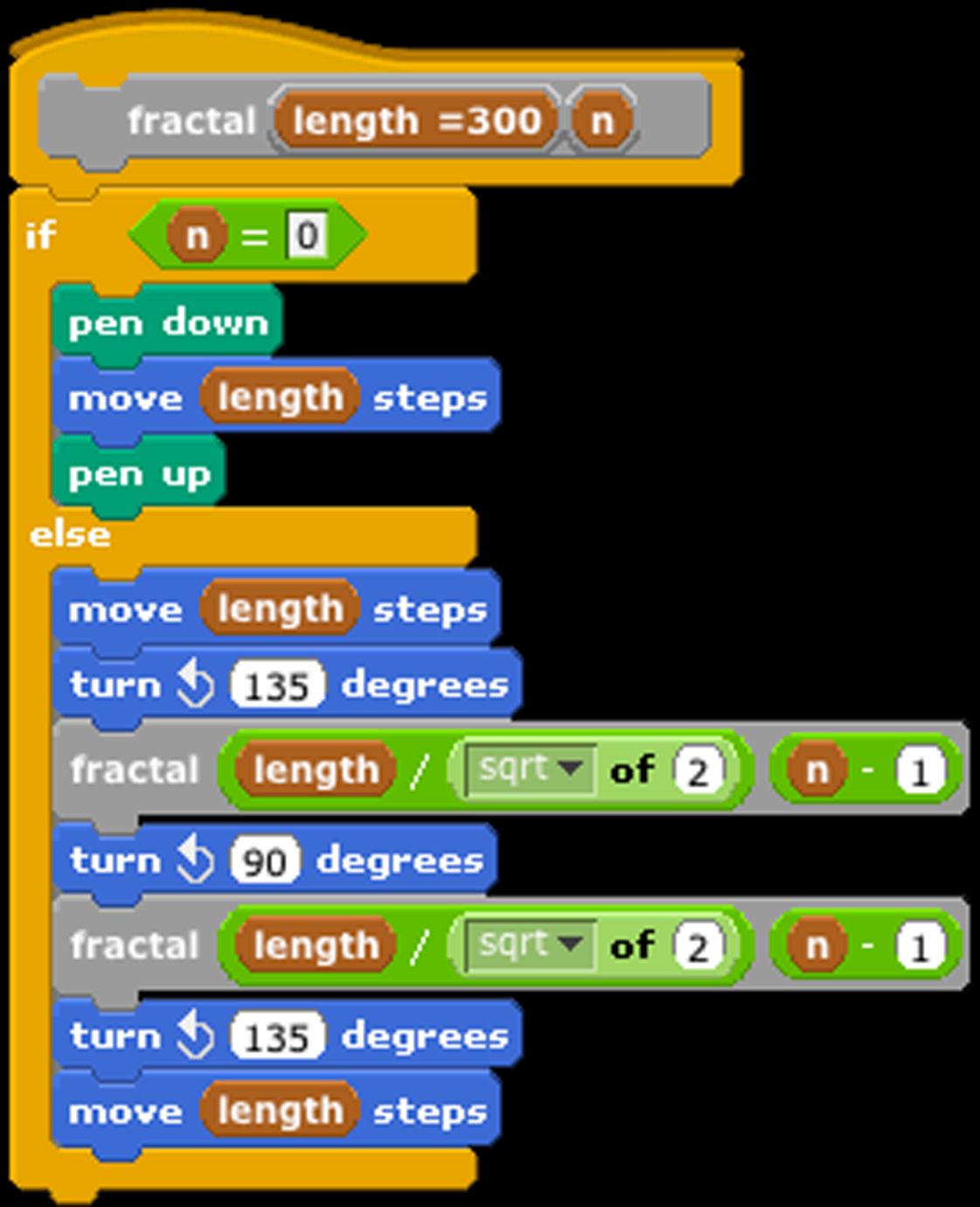
Concurrency

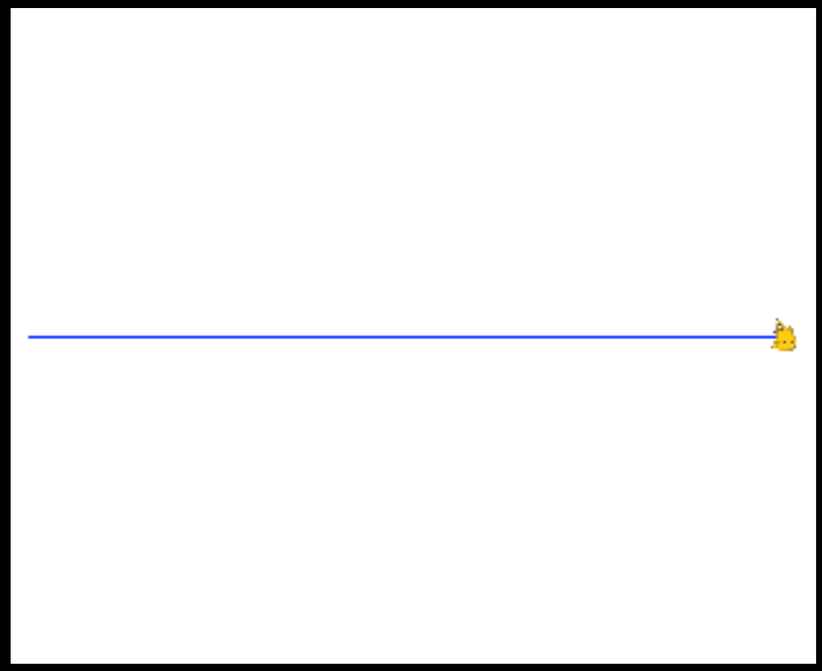
List adds up to 200?

No! Why?

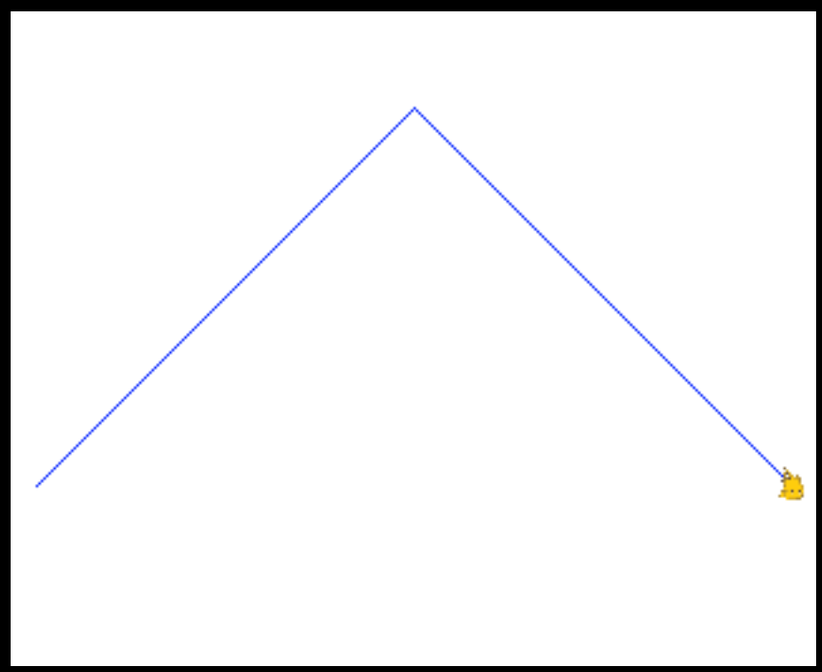
No "wait" here means these might access the same indexes (or not access them at all), by interrupting a broadcast script that is already running!

Recursion





n=0



n=1

Recursion

?

n=2

?

n=4

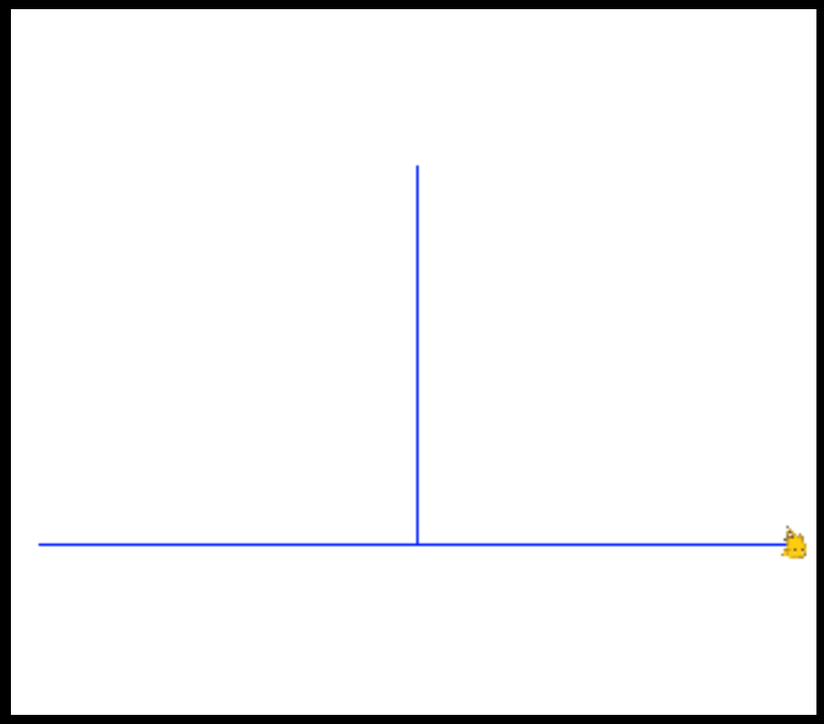
?

n=3

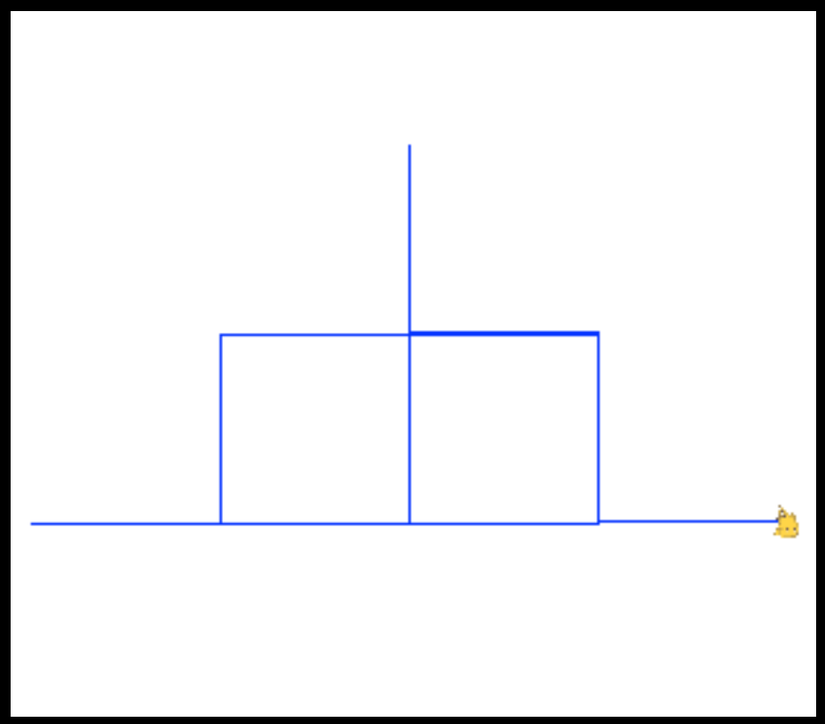
?

n=∞

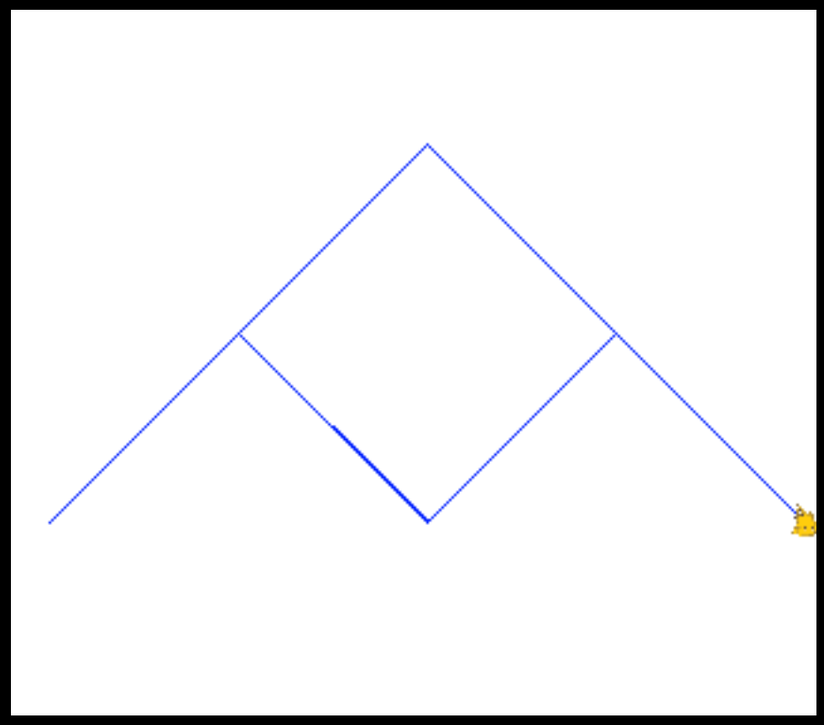
Recursion



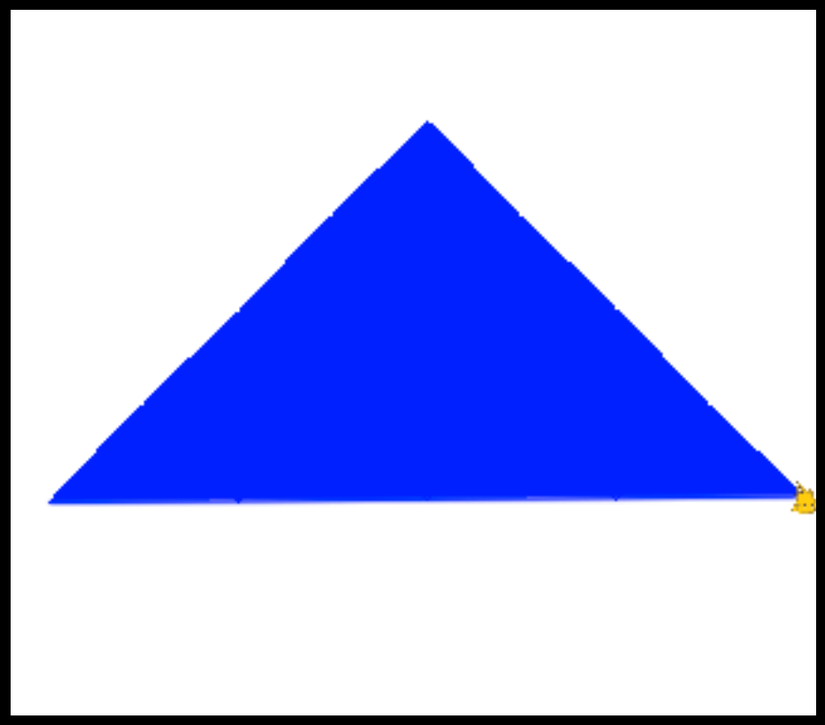
n=2



n=4



n=3



n=∞

Recursion

An algorithmic technique where a function, in order to accomplish a task, calls itself with some part of the task.

- Recursive solutions involve two major parts:
  1. **Base case**(s), in which the problem is simple enough to be solved directly,
  2. **Recursive case**(s). A recursive case has three components:
     1. **Divide the problem** into one or more simpler or smaller parts of the problems,
     2. **Invoke the function** (recursively) on each part, and
     3. **Combine the solutions** of the parts into a solution for the problem.

- Depending on the problem, any of these may be trivial or complex.

http://inst.eecs/~cs3l/sp09/lectures/L06/2009SpCS3L06.html

Data Structure : Hash Tables

Can we use “MapReduce” to build a hash table?

(we’ll come back to that...)

Lambdas & HOFs

- What is a Lambda?

- What is a Higher-Order Function (HOF)?

Lambdas & HOFs

- A "First Class" Procedure
- "Function as Data"

Lambdas & HOFs

- What is a Higher-Order Function (HOF)?
  - In mathematics and computer science, **higher-order functions**, **functional forms**, or **functionals** are functions which do *at least one* of the following:

- **take one or more functions as an input**
- **output a function**

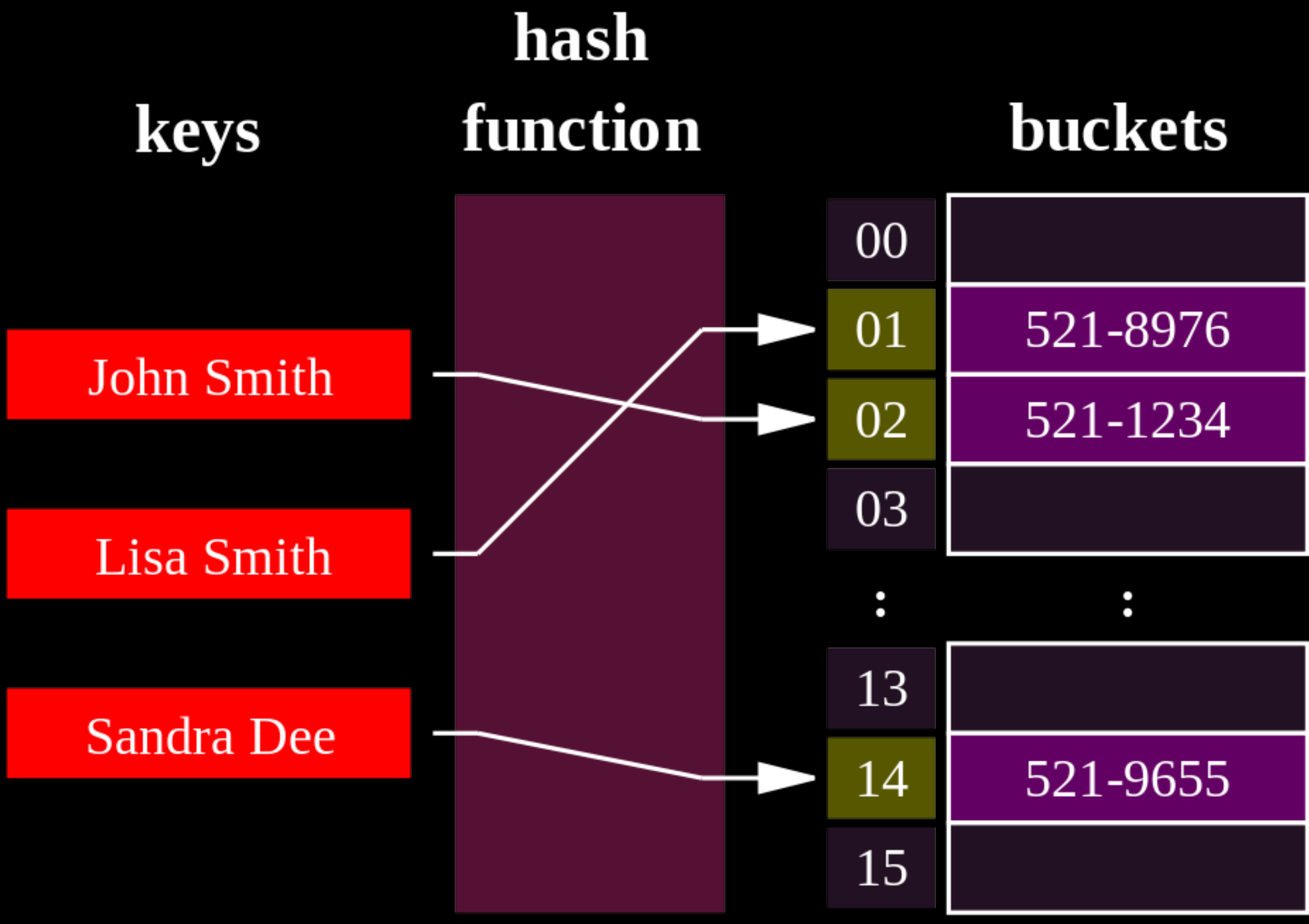
http://en.wikipedia.org/wiki/Higher-order_function

Lambdas & HOFs

- Useful HOFs (you can build your own!)
  - map `Reporter` over `List`
    - Report a new list, every element `E` of `List` becoming `Reporter(E)`
  - keep items such that `Predicate` from `List`
    - Report a new list, keeping only elements `E` of `List` if `Predicate(E)`
  - combine with `Reporter` over `List`
    - Combine all the elements of `List` with `Reporter(E)`
    - This is also known as “reduce”
- Acronym example
  - keep ➔ map ➔ combine

http://inst.eecs.berkeley.edu/~cs10/sp11/lec/17/src/2011-03-30-CS10-L17-DG-HOF-I.pptx

Data Structure : Hash Tables



Data Structure : Hash Tables

- Determine size of hash table.
- Hashing function (algorithm) to generate keys (index) from values (items).
- Modulo key (index) by hash table size.
- Store key and value in hash table at the resulting index.
- Find key in table using above algorithms (reading value instead of storing it).

http://en.wikipedia.org/wiki/Hash_table

Data Structure : HOF/Hash Tables

Can we use “MapReduce” to build a hash table?

Data Structure : HOF/Hash Tables

Yes!

- Try to program one on your own... it's *great* practice with HOFs, Lists, etc.!
- I will post the solution to Piazzza...