CS1010

Programming Methodology

Lecture 2

21 August 2018

Admin Matters
Unit 3: **Functions**
Unit 4: **Types**

Q&A

https://piazza.com/class/jkqlna92ju045j

Important Dates

October 6, 2018 (PE1) 0900 - 1200

November 10, 2018 (PE2) 1300 - 1600

Accounts

- Register for an SoC UNIX account if you do not have one at https://mysoc.nus.edu.sg/~newacct/

- Register for a GitHub account if you do not have one at https://www.github.com/

- Activate your Piazza account (link emailed to you)

Tutorial starts next week

Balloting / Allocation Issues ?

contact Mr Chow Yuan Bin

chowyb@comp.nus.edu.sg

Tutorial 1

Problem Set 1
UNIX walkthrough

Activate your access to the SoC computer clusters (https://mysoc.nus.edu.sg/~myacct/services.cgi), which include the CS1010 programming environment (PE).

CS1010 PE Hosts

pe111.comp.nus.edu.sg

pe112.comp.nus.edu.sg

..

pe120.comp.nus.edu.sg

Access them via ssh (secure shell)

Mac / Linux: command line
Windows: XShell

Try out the UNIX commands:

https://nus-cs1010.github.io/1819-s1/unix/

If you want to get ahead (Tutorial 2), learn `vim` with `vimtutor`

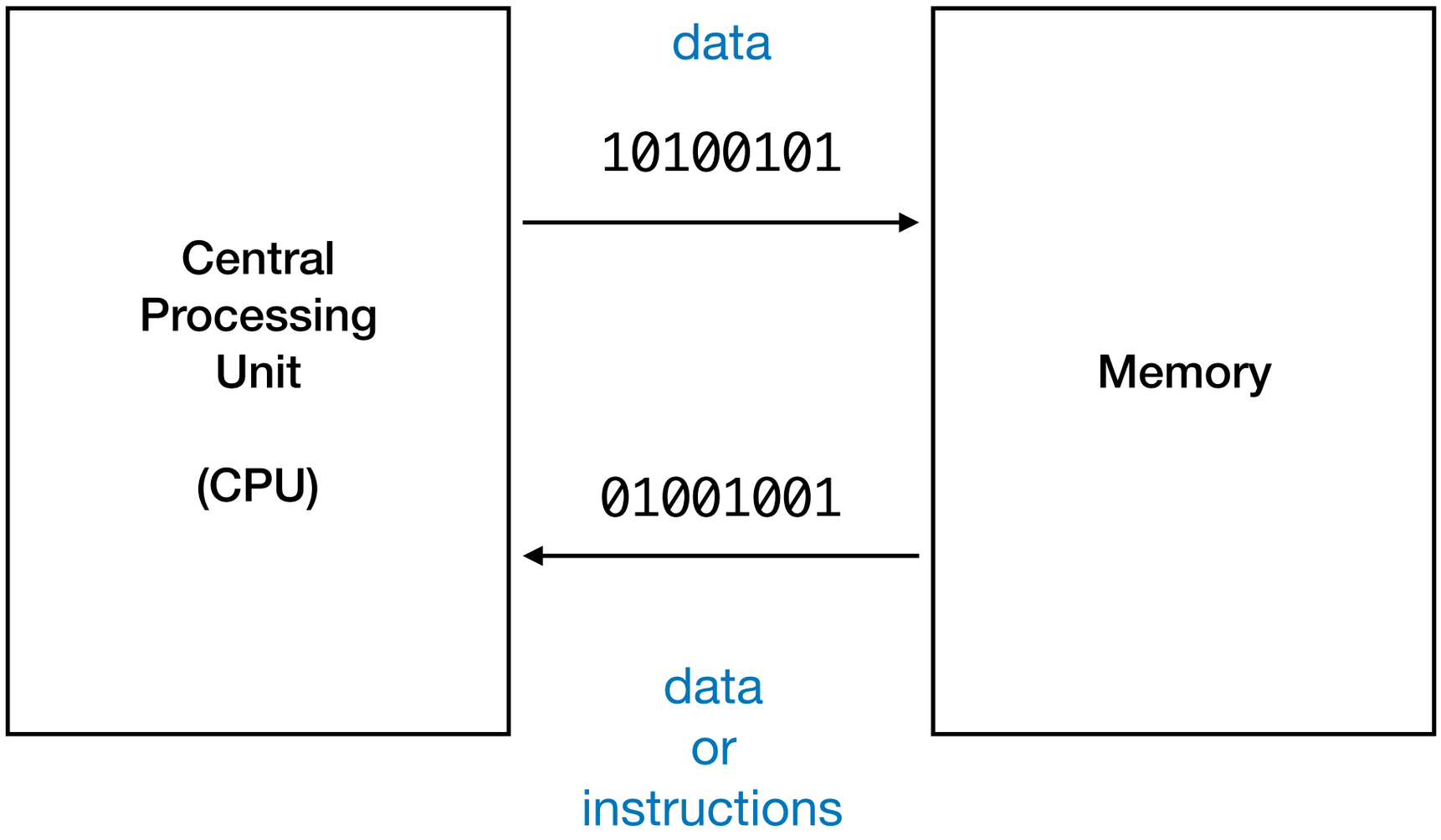
Why command line interface (CLI)?

Why `vim`?

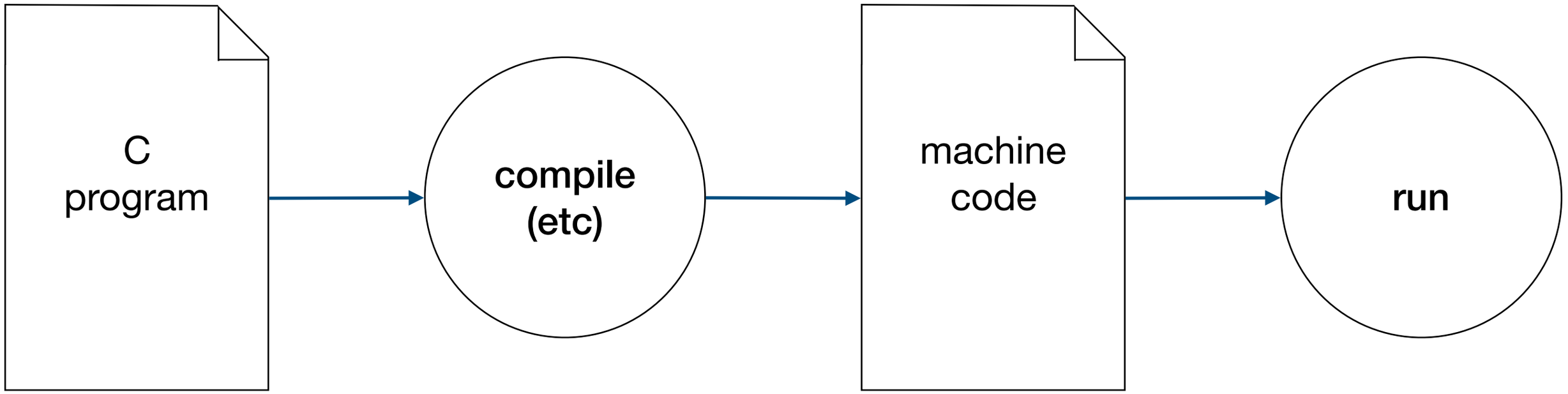
Previously, on

CS1010

A Simplified View of a Computer



Compiling High-Level Program to Machine Code



Learning to write a program that does what you want it to do is actually not difficult.

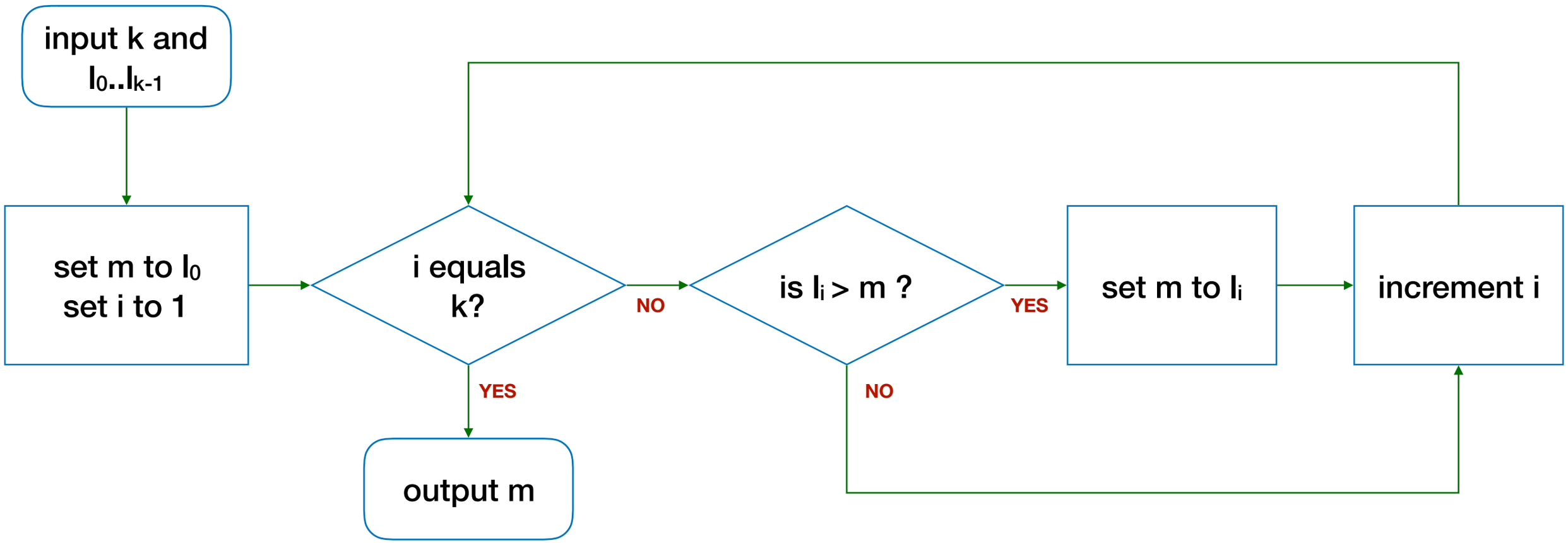
Knowing what you want your program to do is the more challenging part!

Computational Problems

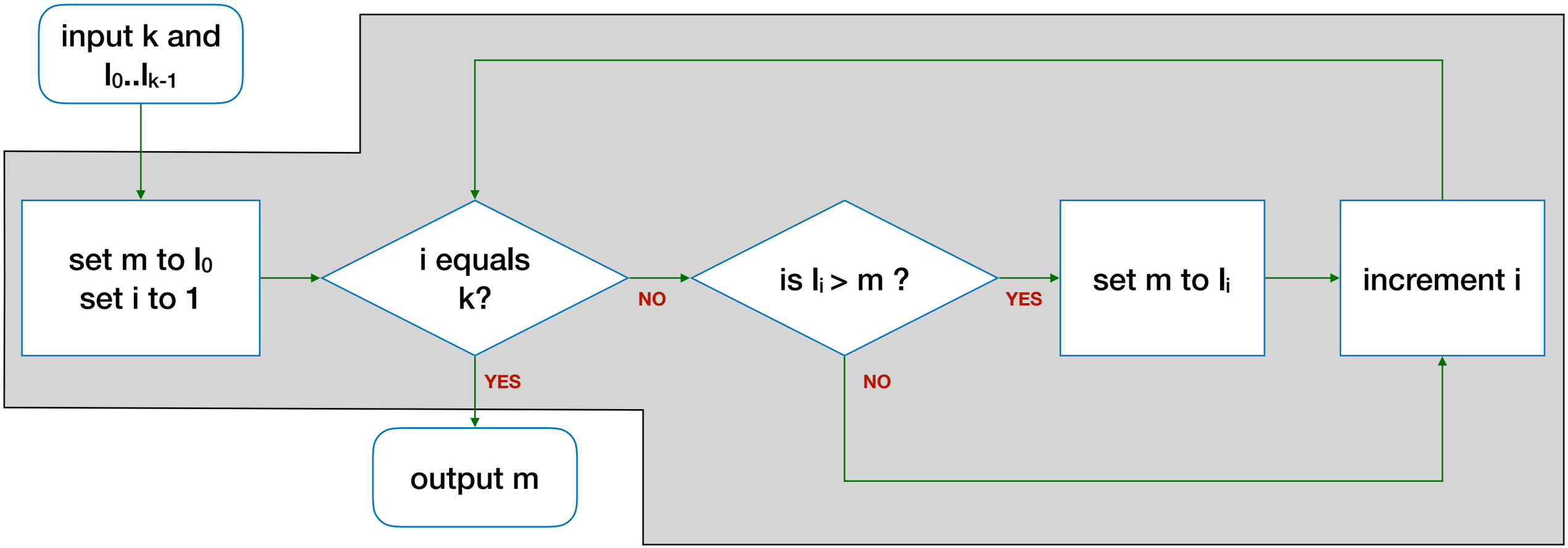
- Problems that can be solved step-by-step by a computer
- Have well-defined inputs, outputs, and constraints

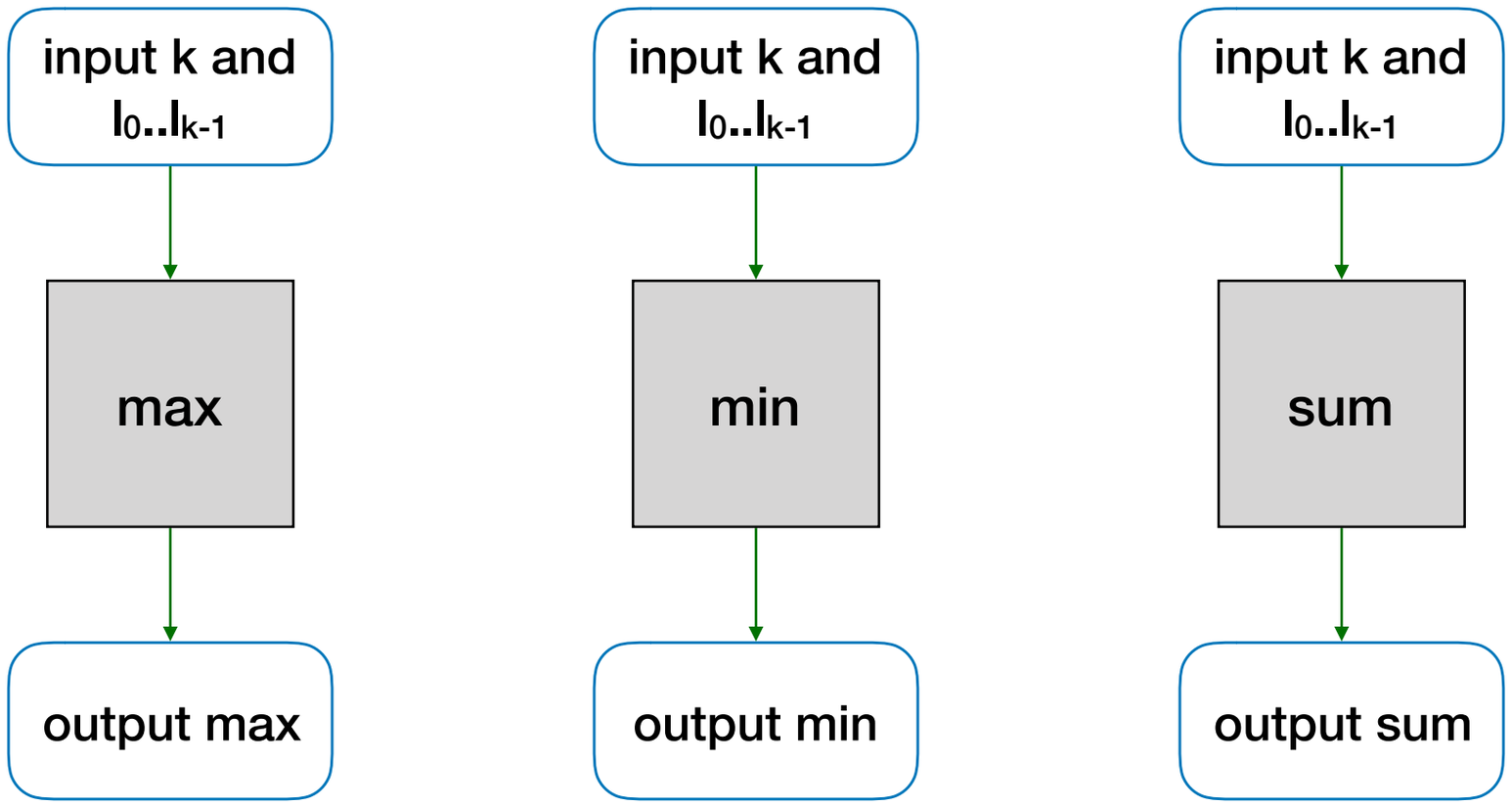
The steps to solve a computational problem are called “algorithm”

Flowchart


input k and
$l_0..l_{k-1}$
set m to l_0
set i to 1
i equals
k?
YES
output m
NO
is l_i > m ?
YES
set m to l_i
increment i
NO

Functions


input k and
$l_0..l_{k-1}$
set m to l_0
set i to 1
i equals
k?
NO
YES
output m
is l_i > m ?
YES
NO
set m to l_i
increment i



max(L, k)

min(L, k)

sum(L, k)

Find the range

- The range of a given list of k integers is the difference between the max and the min values in the list.
- Give an algorithm to find the range

$$\max(L, k) - \min(L, k)$$

Refer to a previous solution to a sub-problem, which we assume we already know how to solve.

“Wishful Thinking”

Powerful tool to help us think at a higher level

Can worry about
what a function does.

Not **how** it does it.

A C program is just a collection of functions.

Functions written by you

Functions provided by the standard C library or other libraries

Find the mean

- Give an algorithm to find the mean value in a given list L of k integers.

sum(L, k) / k

Find the Std Dev

- Give an algorithm to find the standard deviation of a given list L of k integers.

$$\sqrt{\frac{\sum_{i=0}^{k-1}(l_i - \mu)^2}{k}}$$

Break it down to subproblems

$$\sqrt{\frac{\sum_{i=0}^{k-1}(l_i - \mu)^2}{k}}$$

$$\sum_{i=0}^{k-1}(l_i - \mu)^2$$

set *mu* to *mean(L, k)*
set *L’* to *subtract(L, k, mu)*
set *L”* to *square(L’, k)*
set *total* to *sum(L”, k)*

$$\sum_{i=0}^{k-1}(l_i - \mu)^2$$

sum(square(subtract(L, k, mean(L, k)), k), k)

$$\sum_{i=0}^{k-1}(l_i - \mu)^2$$

set *total* to

sum(square(subtract(L,k,mean(L,k)),k),k)

$$\frac{\sum_{i=0}^{k-1}(l_i-\mu)^2}{k}$$

mean(square(subtract(L,k,mean(L,k)),k),k)

$$\sqrt{\frac{\sum_{i=0}^{k-1}(l_i-\mu)^2}{k}}$$

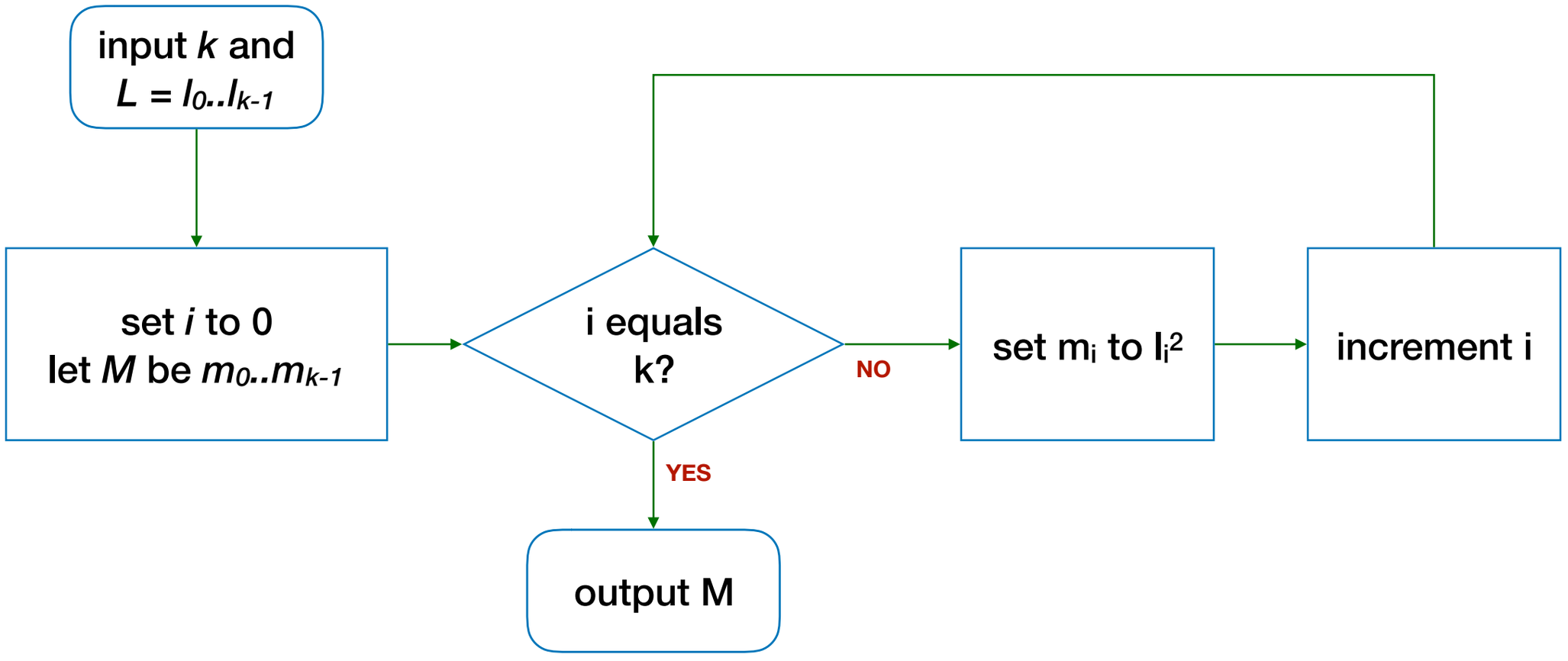
sqrt(mean(square(subtract(L,k,mean(L,k)),k),k))

“Wishful Thinking”

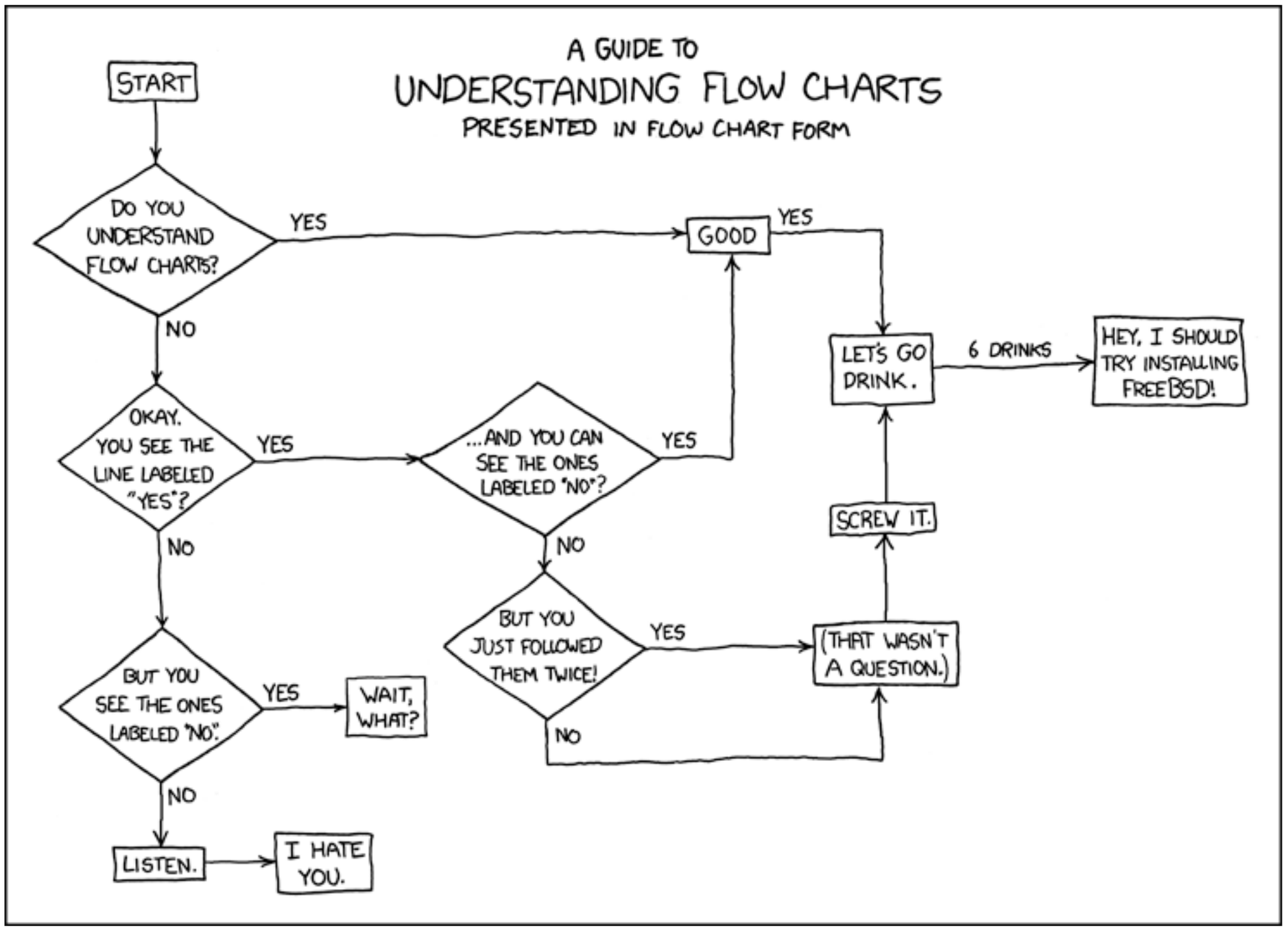
square(L, k)

subtract(L, k, mu)

sqrt(x)


input *k* and
L = $l_0..l_{k-1}$
set *i* to 0
let *M* be $m_0..m_{k-1}$
i equals
k?
NO
YES
set m_i to l_i^2
increment i
output M

https://xkcd.com/518/



Find the maximum number

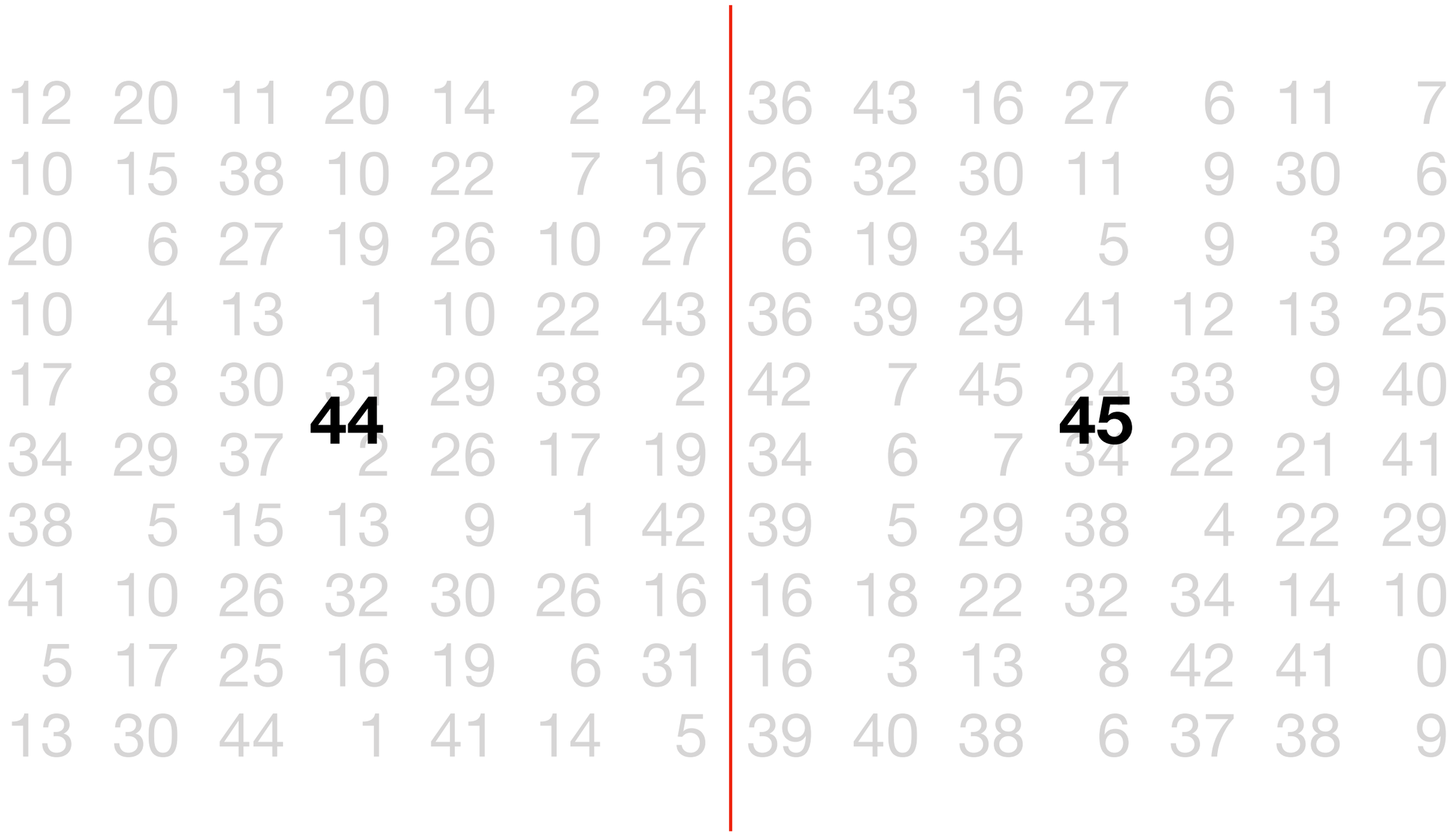
5 9 8

1 3 2

Find the maximum

12	20	11	20	14	2	24	36	43	16	27	6	11	7
10	15	38	10	22	7	16	26	32	30	11	9	30	6
20	6	27	19	26	10	27	6	19	34	5	9	3	22
10	4	13	1	10	22	43	36	39	29	41	12	13	25
17	8	30	31	29	38	2	42	7	45	24	33	9	40
34	29	37	2	26	17	19	34	6	7	34	22	21	41
38	5	15	13	9	1	42	39	5	29	38	4	22	29
41	10	26	32	30	26	16	16	18	22	32	34	14	10
5	17	25	16	19	6	31	16	3	13	8	42	41	0
13	30	44	1	41	14	5	39	40	38	6	37	38	9

Find the maximum



Find the maximum

12	20	11	20	14	2	24	36	43	16	27	6	11	7
10	15	38	10	22	7	16	26	32	30	11	9	30	6
20	6	27	19	26	10	27	6	19	34	5	9	3	22
10	4	13	1	10	22	43	36	39	29	41	12	13	25
17	8	30	31	29	38	2	42	7	45	24	33	9	40
34	29	37	2	26	17	19	34	6	7	34	22	21	41
38	5	15	13	9	1	42	39	5	29	38	4	22	29
41	10	26	32	30	26	16	16	18	22	32	34	14	10
5	17	25	16	19	6	31	16	3	13	8	42	41	0
13	30	44	1	41	14	5	39	40	38	6	37	38	9

Find the maximum

52

20	11	20	14	2	24	36	43	16	27	6	11	7
10	15	38	10	22	7	16	26	32	30	11	9	30	6
20	6	27	19	26	10	27	6	19	34	5	9	3	22
10	4	13	1	10	22	43	36	39	29	41	12	13	25
17	8	30	31	29	38	2	42	7	45	24	33	9	40
34	29	37	2	26	17	19	34	6	7	34	22	21	41
38	5	15	13	9	1	42	39	5	29	38	4	22	29
41	10	26	32	30	26	16	16	18	22	32	34	14	10
5	17	25	16	19	6	31	16	3	13	8	42	41	0
13	30	44	1	41	14	5	39	40	38	6	37	38	9

45

“Wishful Thinking”

max(L, k) for smaller *k*

max'(L, i, j) finds the max value for $l_i \ldots l_j$

max'(L, i, j) finds the max value for $l_i \ldots l_j$

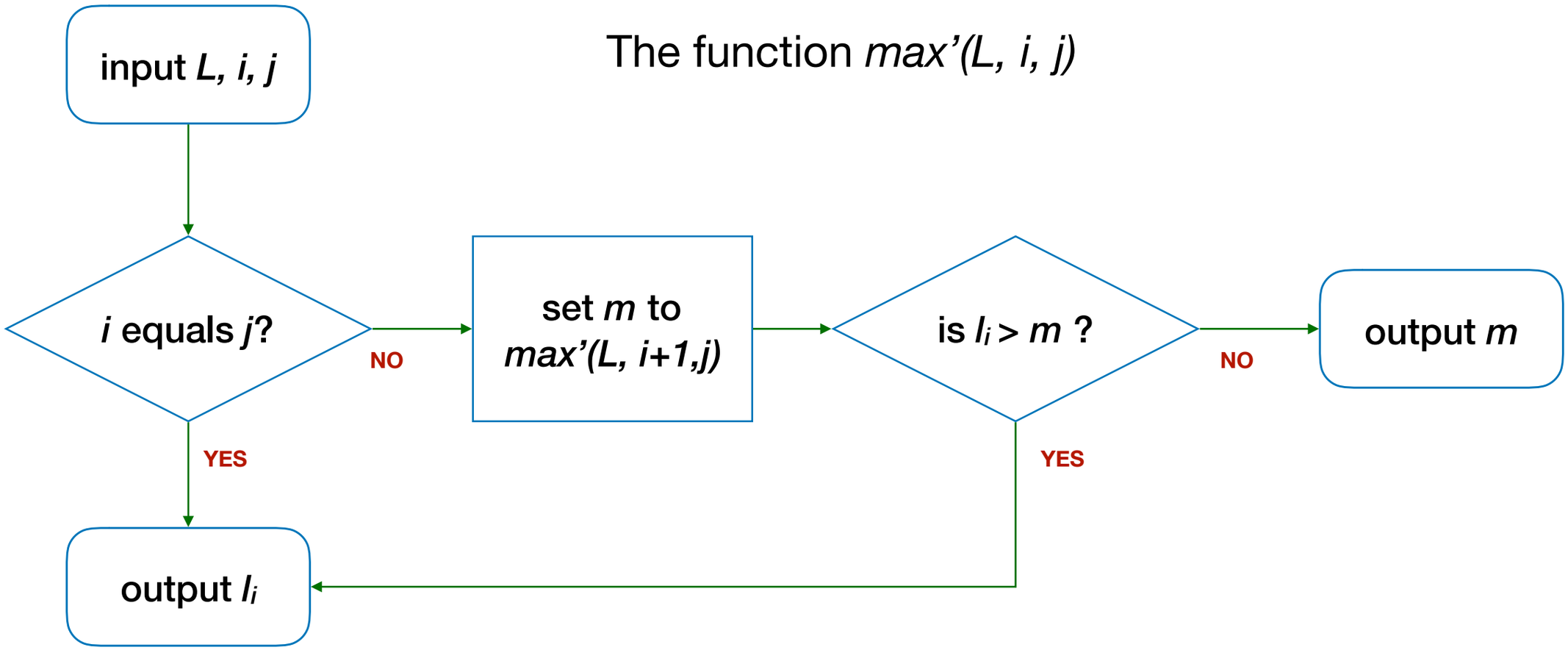
if *i* equals *j*
then l_i is the max

if *i* does not equals *j*

then max is either

l_i or *max'(L, i+1, j)*

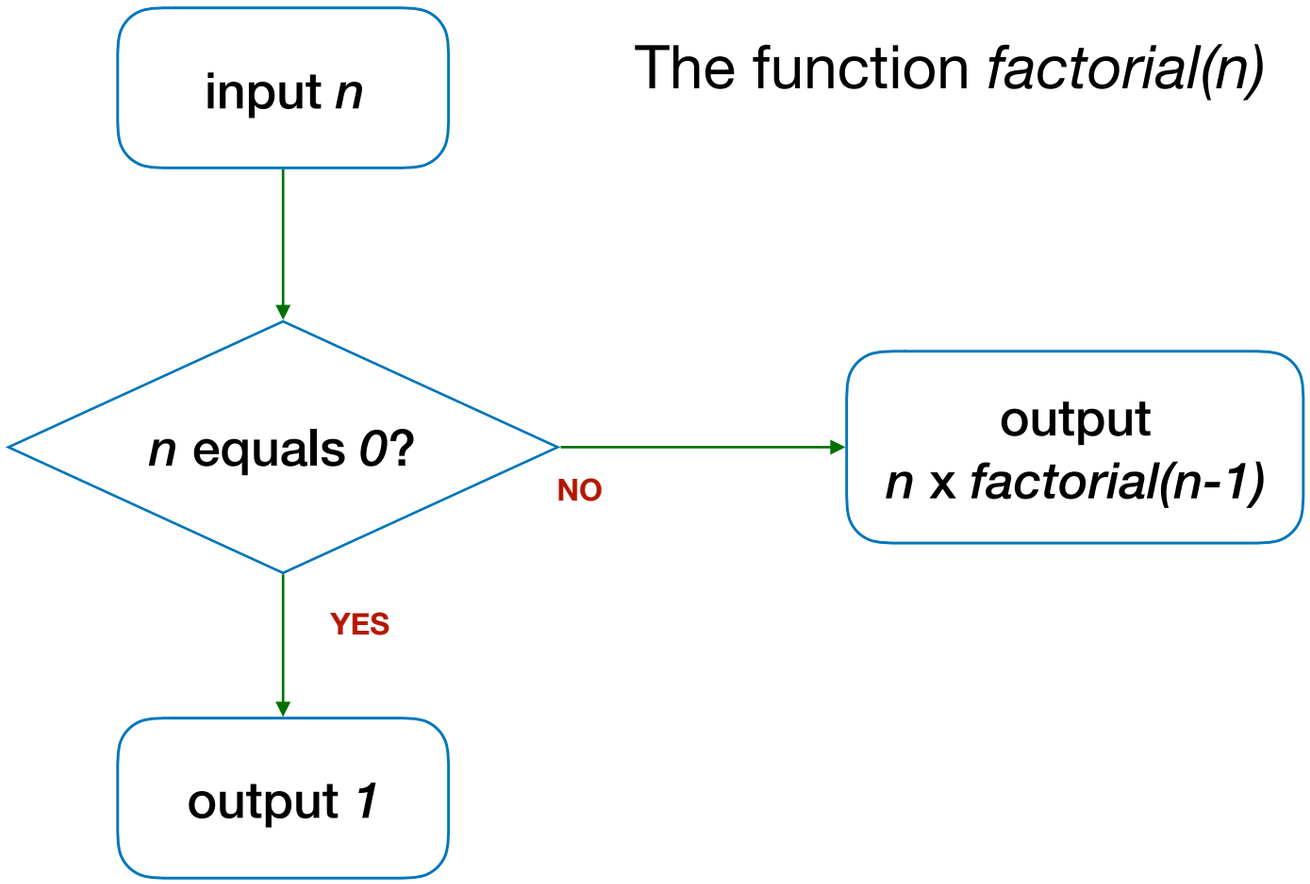
The function *max'(L, i, j)*



Find the Factorial

- Give n, find n!
- n! = n x (n-1) x (n-2) x … 2 x 1
  = n x (n-1)!
- Special case: 0! = 1

The function *factorial(n)*



Recursion

A function calling itself

Tutorial 2

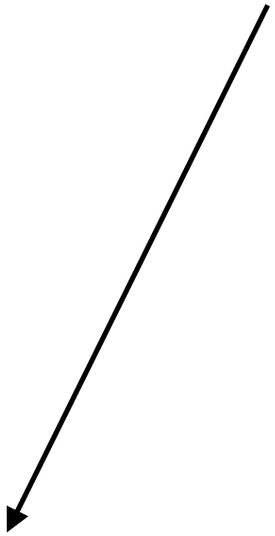
Problem Set 3

`vim`

Warm Up Assignment

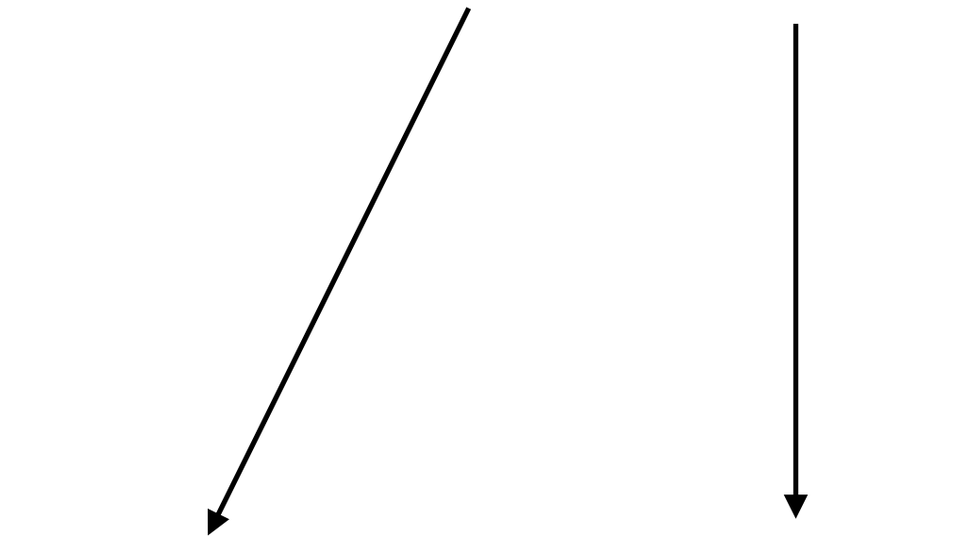
Types

10100101010101010011111010...



1933091

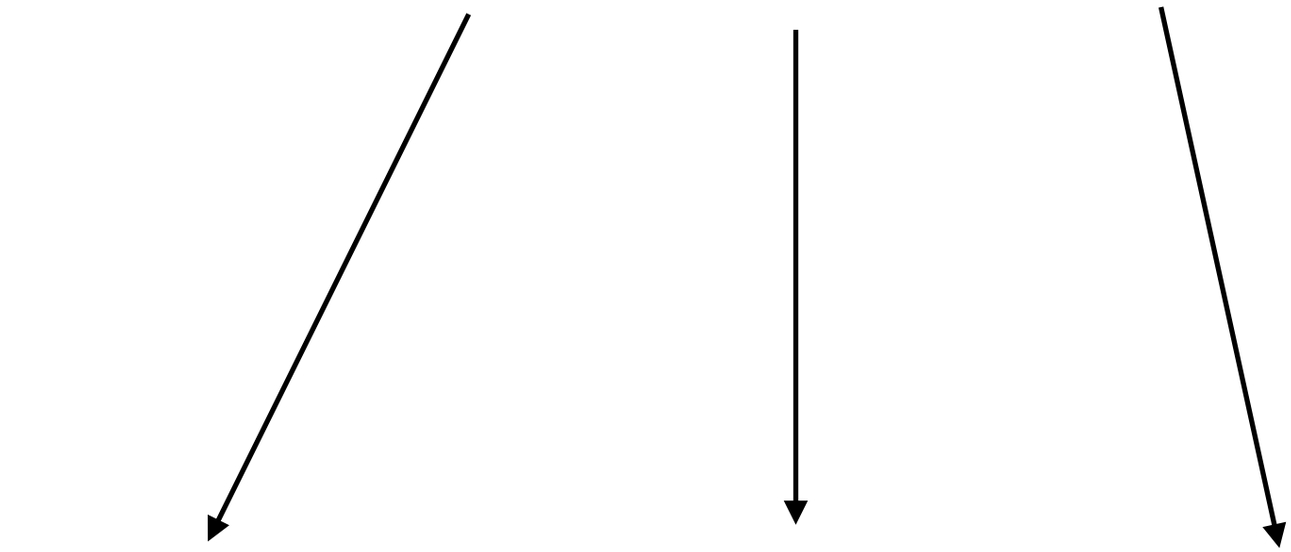
10100101010101010011111010...



1933091

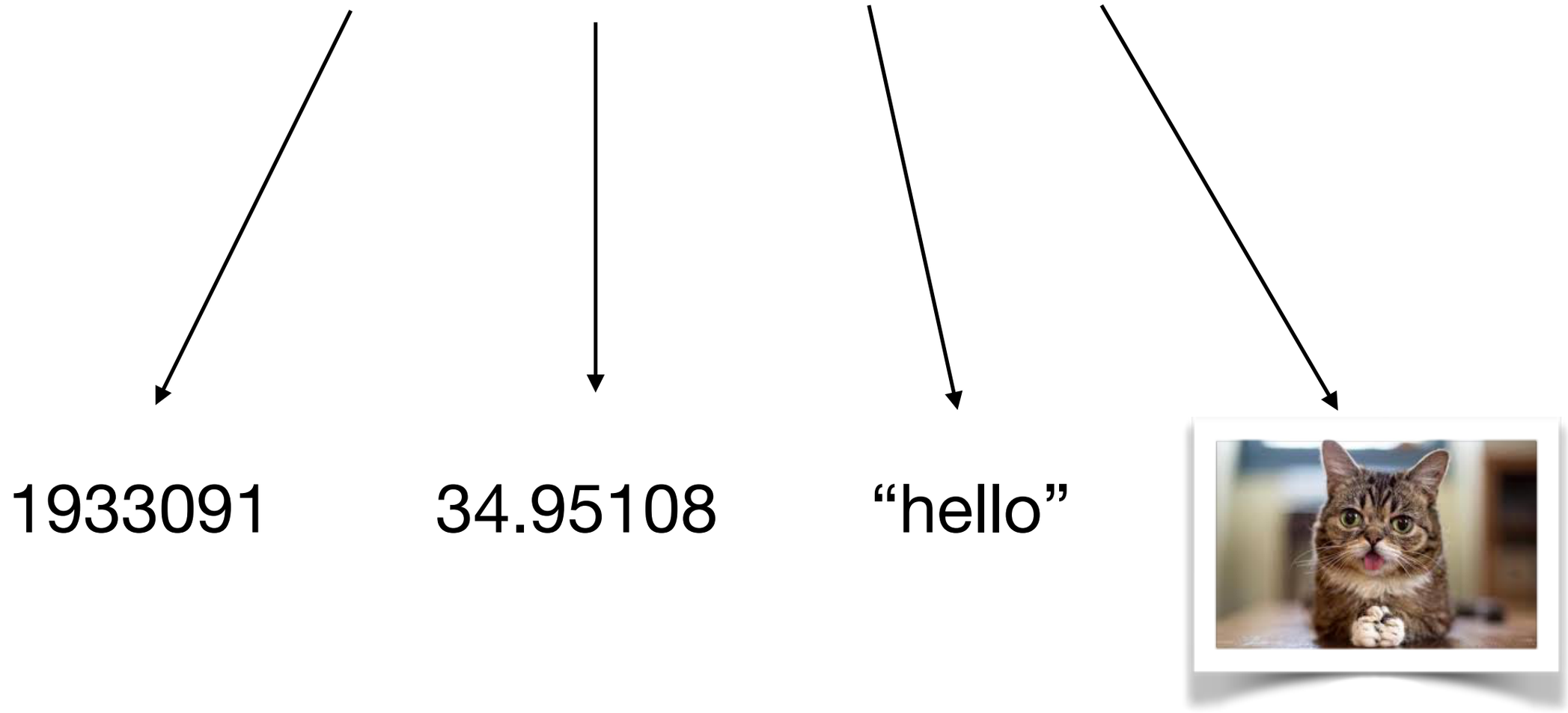
34.95108

10100101010101010011111010...



1933091 34.95108 “hello”

10100101010101010011111010...



1933091 34.95108 “hello”

Variable needs a **type** to be interpreted correctly.

A type is represented
a finite number of bits.

Can be different number
of bits for different types

1 bit: `1` or `0`

black or white

true or false

yes or no

S or U

2 bits: 11 00 01 10

north, south, east, west

spring, summer, fall, winter

In general, k bits can represent 2^k values

more bits -> use more memory can represent bigger values

fewer bits -> less memory, but limited range of values

When you program embedded systems, IoT, sensors, etc. You need to be frugal with memory usage.

Not an issue in CS1010
(we will use 32 or 64 bits)

Integers

8 bits: 256 different integers

0 to 255 (for unsigned)
-128 to 127 (for signed)

64 bits:

-9,223,372,036,854,775,808

to

9,223,372,036,854,775,807

Characters

8 bits: 127 different symbols using ASCII standards

`0-9, A-Z, a-z`

`<>?:”{}!@#$%^&*`

`()_+-=[]\';/.,`

return, tab, escape

etc

up to 32 bits: Unicode

1111101100000000000 😀

Real Numbers

(also called floating point numbers due to how it is represented)

There are **infinitely** many but only **finite** number of representations with bits

We normally use 32, 64, or 128 bits to represent real numbers.

Variable need a **type** to be interpreted correctly.

set *total* to *sum(L, k)*
set *avg* to *mean(L, k)*

k is an integer
total is an integer
avg must be a real number