DEFINING "COMPUTATION"

DISCRETE MATHEMATICS AND THEORY 2

MARK FLORYAN

GOALS!

1. What exactly is “computation”. What exactly do we mean by “computation”? Let’s define these things clearly and explicitly.

2. What is the ***birds-eye view*** of theory of computation (for our course!). Are there different ways to “compute” that have strengths and weaknesses?

3. What ”Math” do we need in order to start thinking about computation properly, and why?

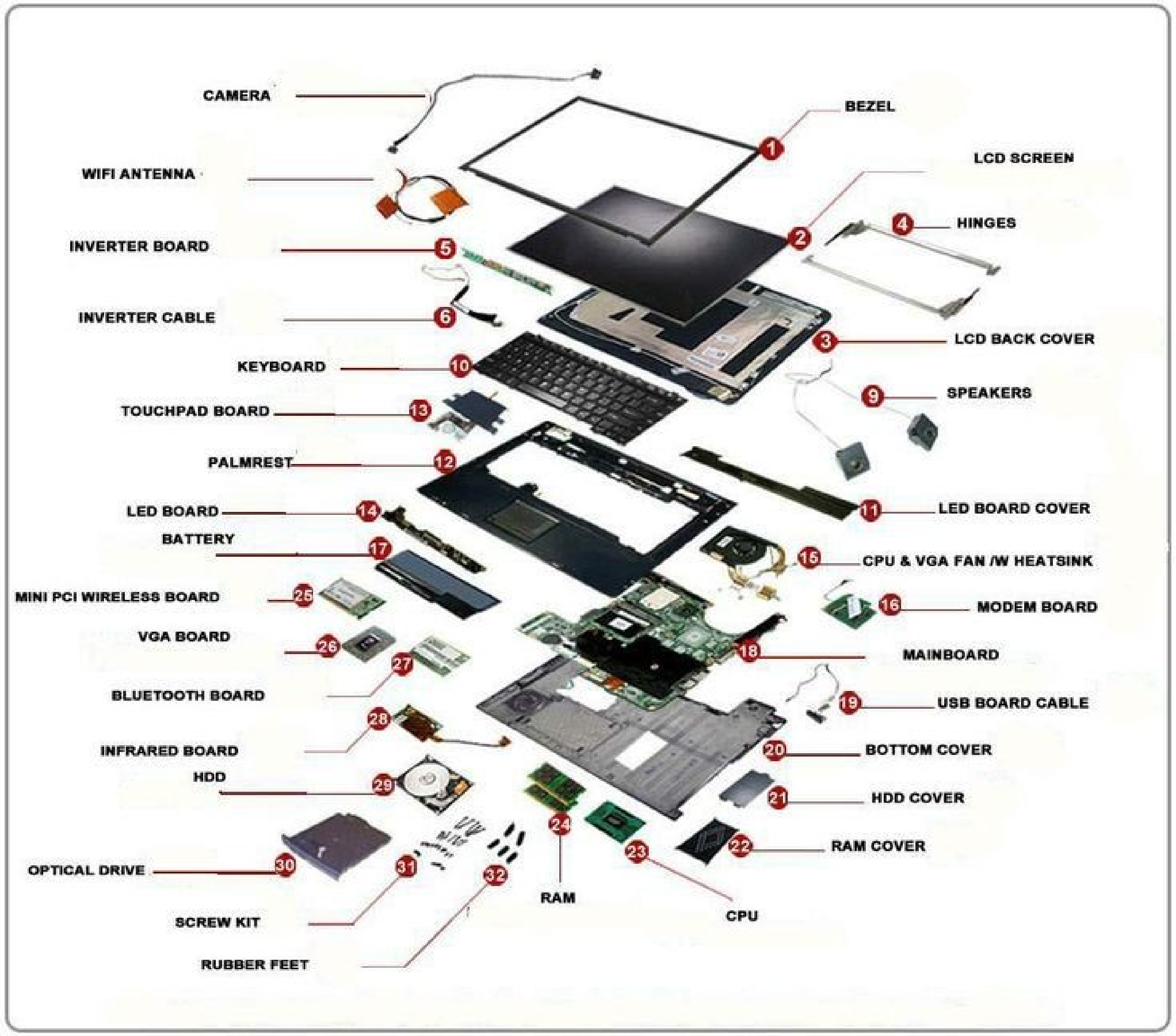
PART 1: WHAT IS COMPUTATION? WHAT IS A COMPUTER?

WHAT IS A COMPUTER?

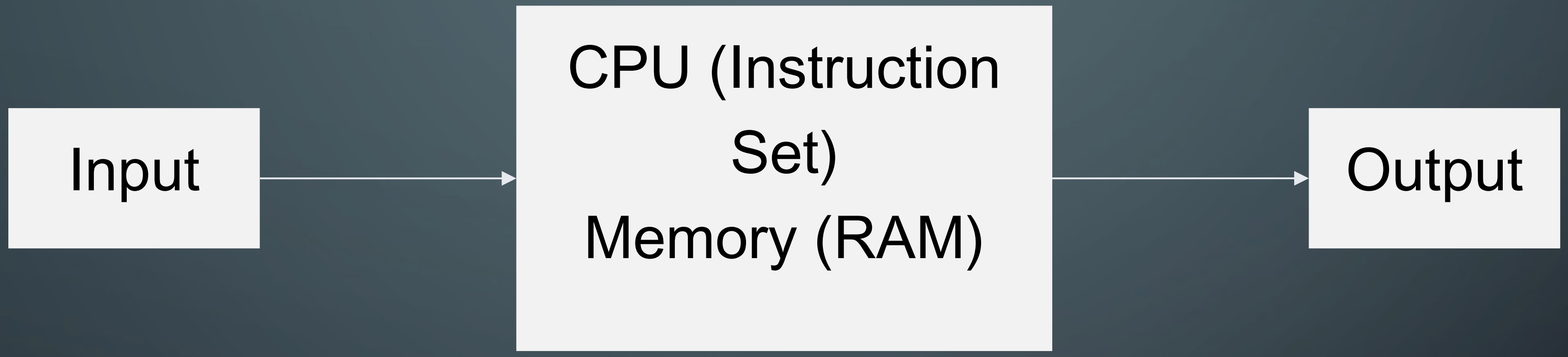
Class discussion:

What is a computer? What do you think?

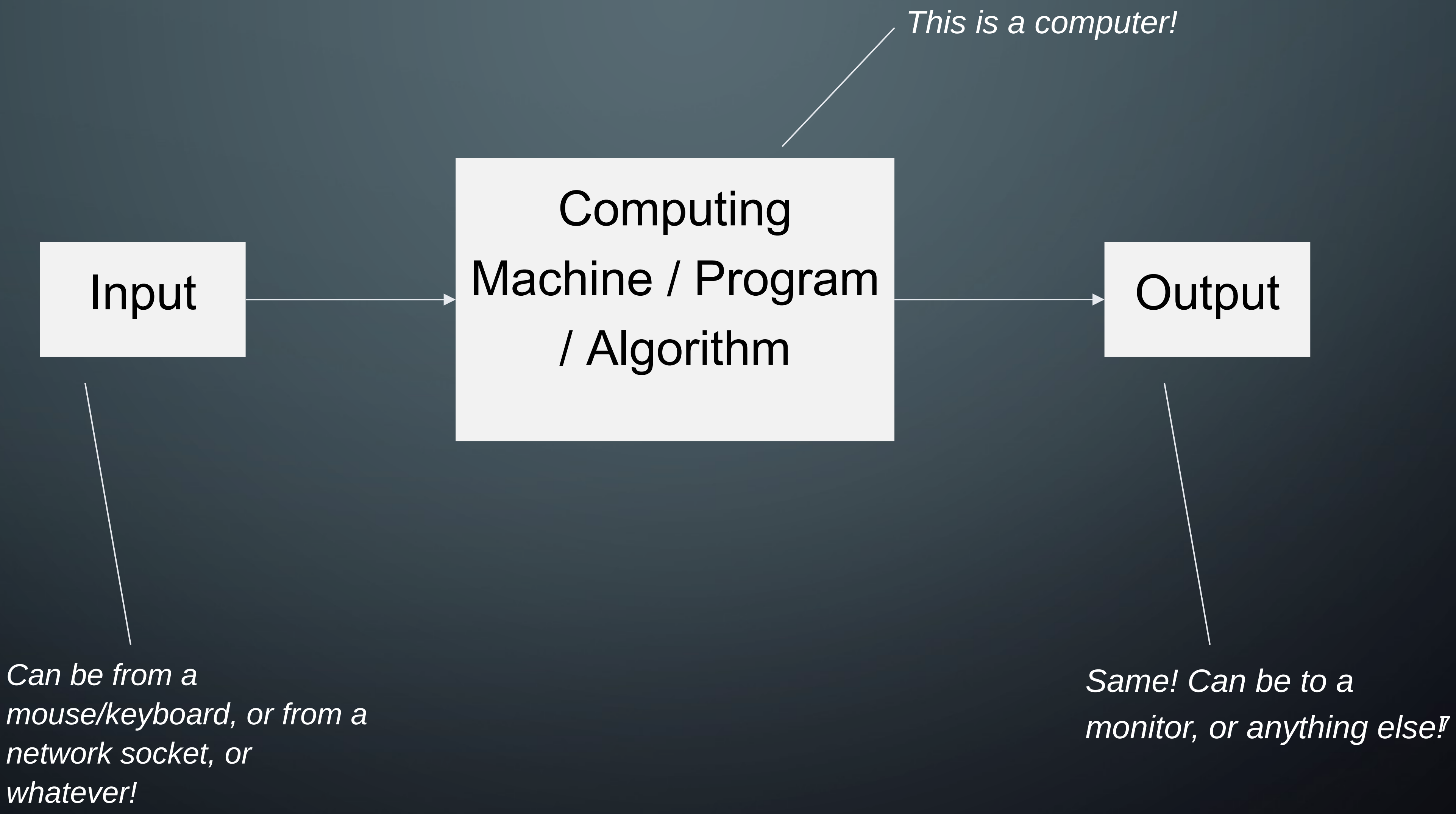
DISCUSSION: WHAT PARTS ARE NECESSARY AT MINIMUM?



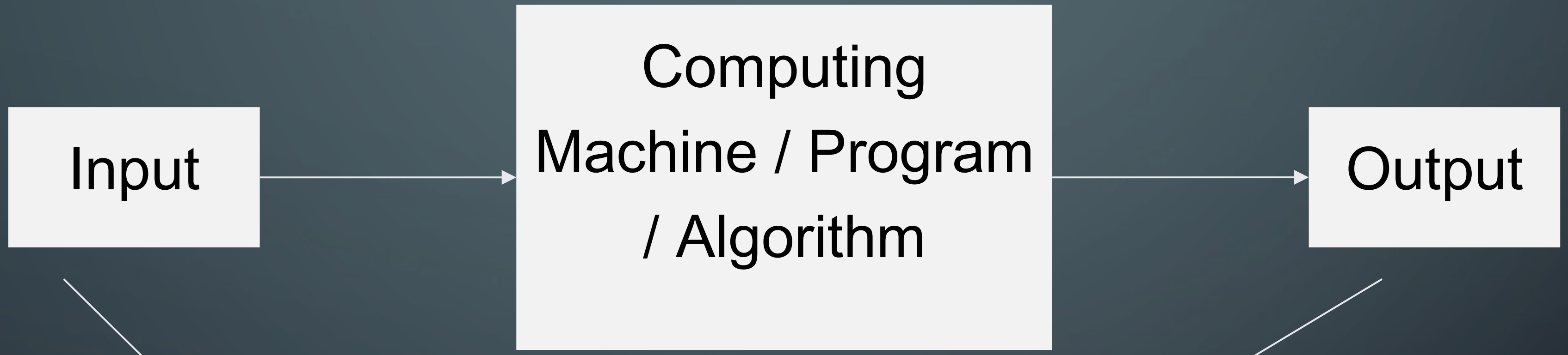
MY ANSWER



MY ANSWER

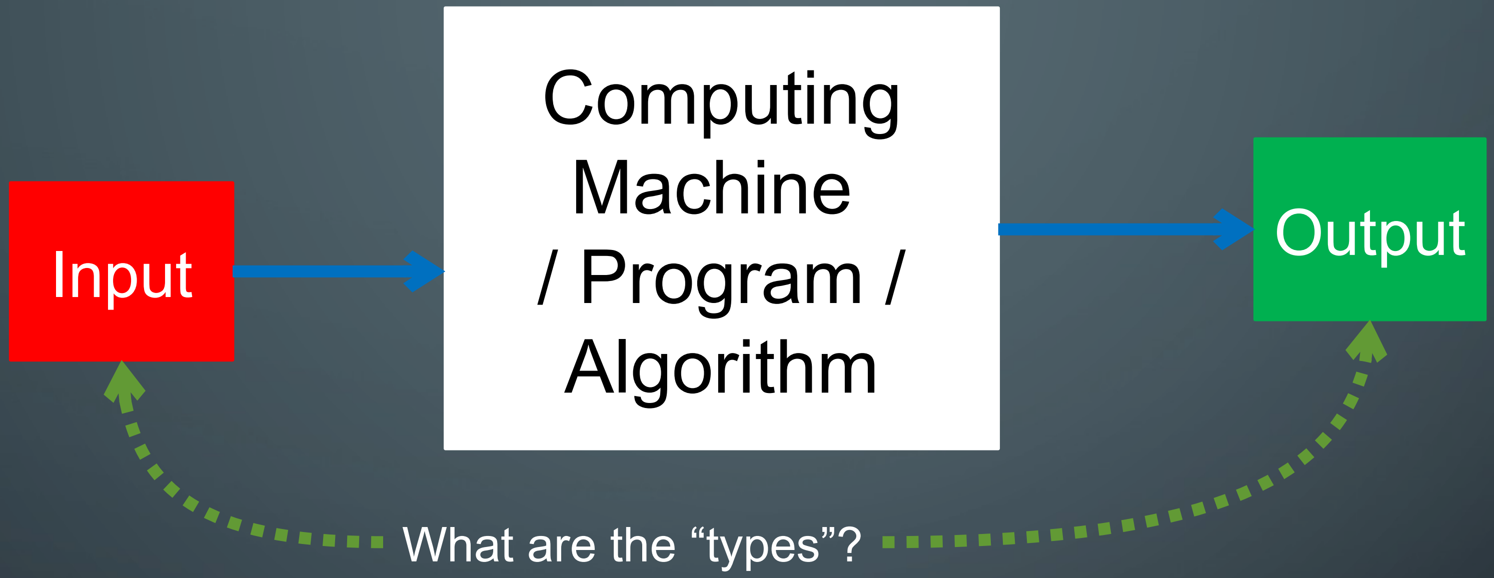

This is a computer!
Input
Computing Machine / Program / Algorithm
Output
Can be from a mouse/keyboard, or from a network socket, or whatever!
Same! Can be to a monitor, or anything else!

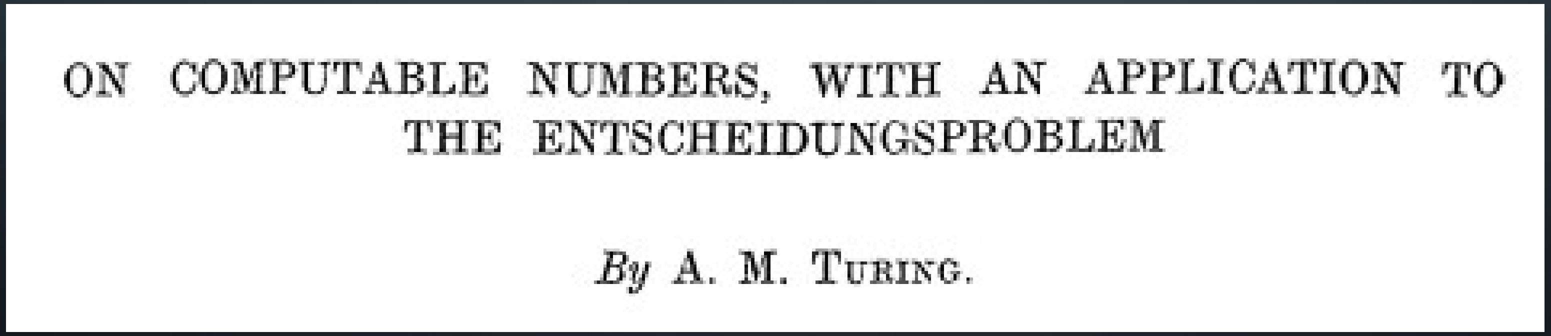
DEFINING INPUT AND OUTPUT



Let's focus on these two first!!

DEFINING INPUT AND OUTPUT

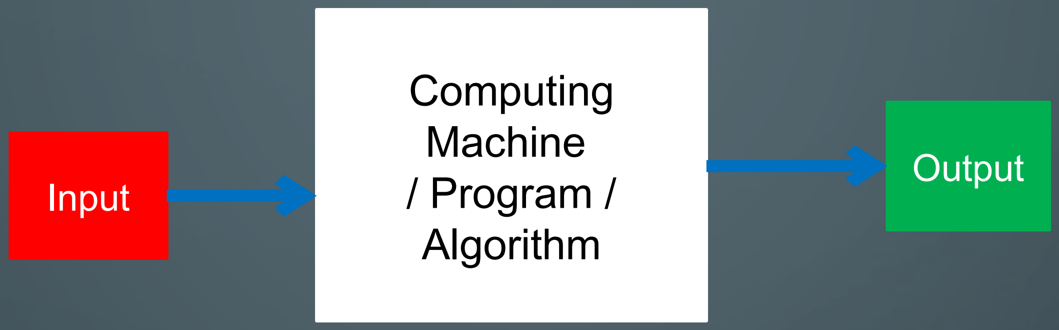



ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

What we compute on: representations of things (e.g. numbers)

DEFINING INPUT AND OUTPUT



- Input and output are ***strings***
- Black box is an ***implementation***
- What are we implementing?
  - ***Functions***: Transformations from a set of strings to another set of strings. More on this soon!

DEFINING INPUT AND OUTPUT

An ***Alphabet*** is a finite set of characters

Notation:

(\Sigma in LaTeX)

Examples:

Strings are built up from an ***Alphabet***

If:

Valid Strings:

00abc10

cccccc

a11

DEFINING INPUT AND OUTPUT

We can define structure of a string in various ways. Examples:

Σ^{n}	The set of all length n strings over alphabet
$\{0,1\}^{3}$	The set of 3-bit strings.

Pop Quiz: If , then

DEFINING INPUT AND OUTPUT

The * operator refers to a string of any length, including 0

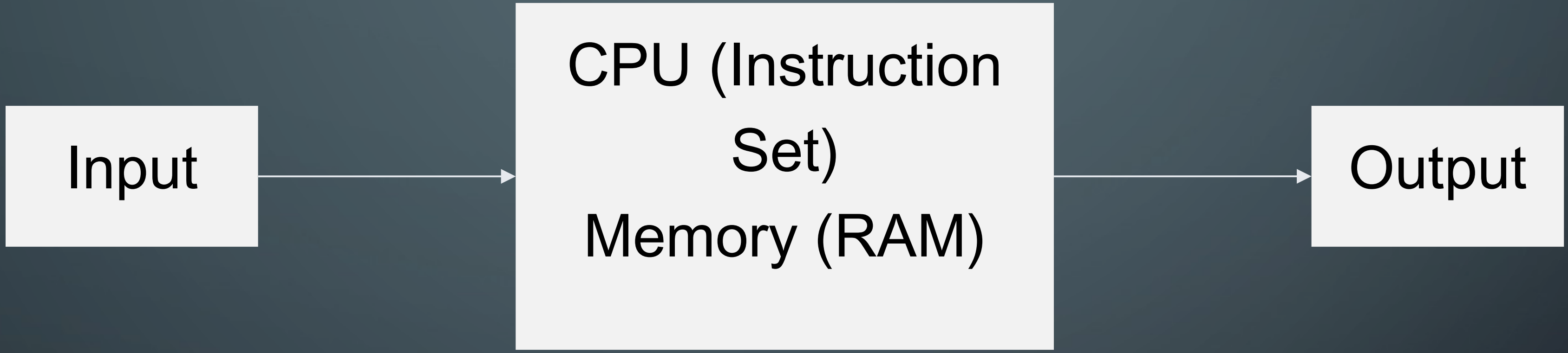
Σ^*	The set of all strings over alphabet of any length (including 0)
$\{0,1\}^*$	

Note that:

Note that:

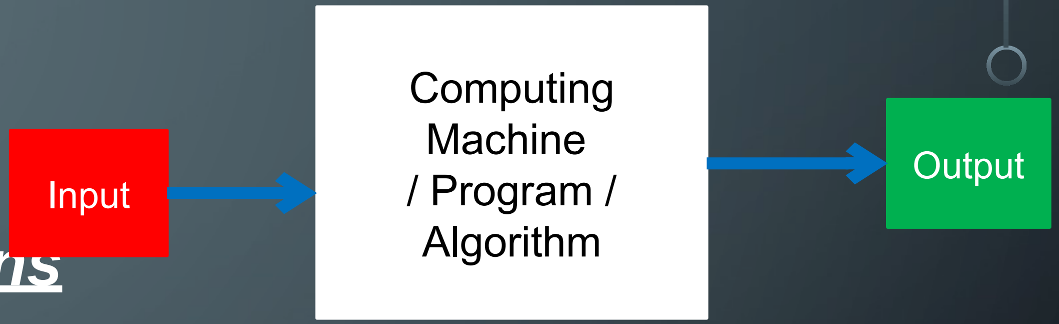
- *The set of three-bit strings*

DEFINING COMPUTATION



Ok! What about this part?

COMPUTING A FUNCTION



- Definition of computation is based on ***functions***
- A function is ***computable*** under a computing model if:
- That model allows for an implementation (way of filling in the black box) such that,
  - For any input (string representing an element from the domain of)
  - The implementation "produces" the correct output

DEFINING COMPUTATION

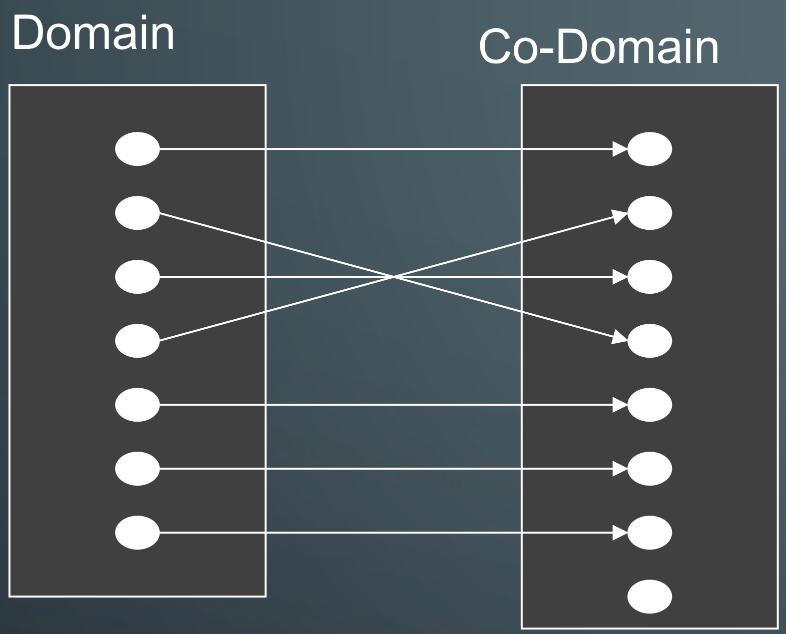
Function: a “mapping” from input to output

- Function maps elements from the set to an element from the set
- : the domain of
- : the co-domain of
- Range/image of :
  - The elements of that are “mapped to” by something

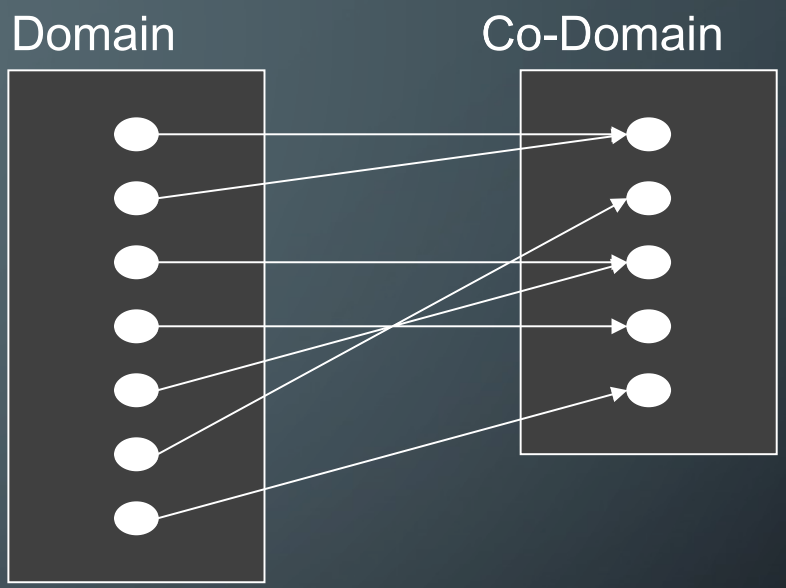
Finite function: a function with a finite domain

is a finite function if is finite. Otherwise it’s an infinite function

INJECTIVE FUNCTIONS



INJECTIVE FUNCTION

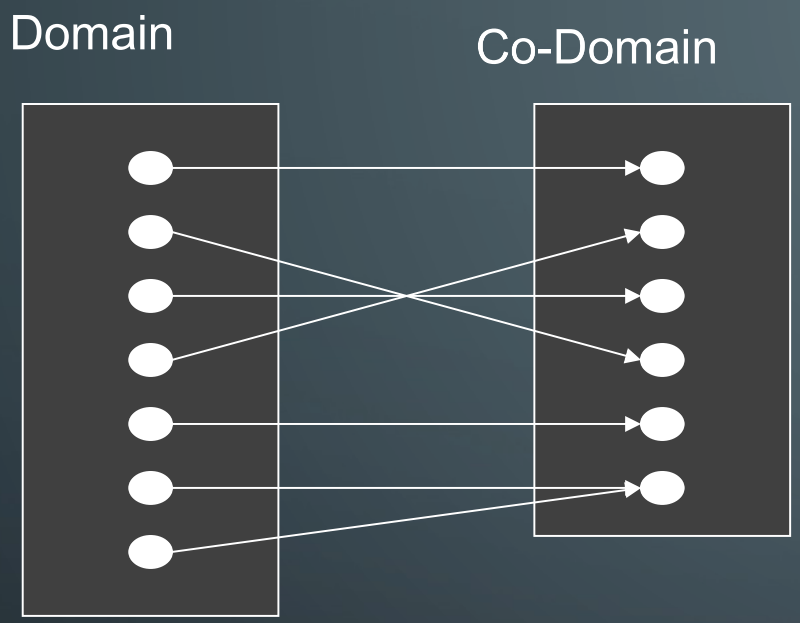


NON-INJECTIVE FUNCTION

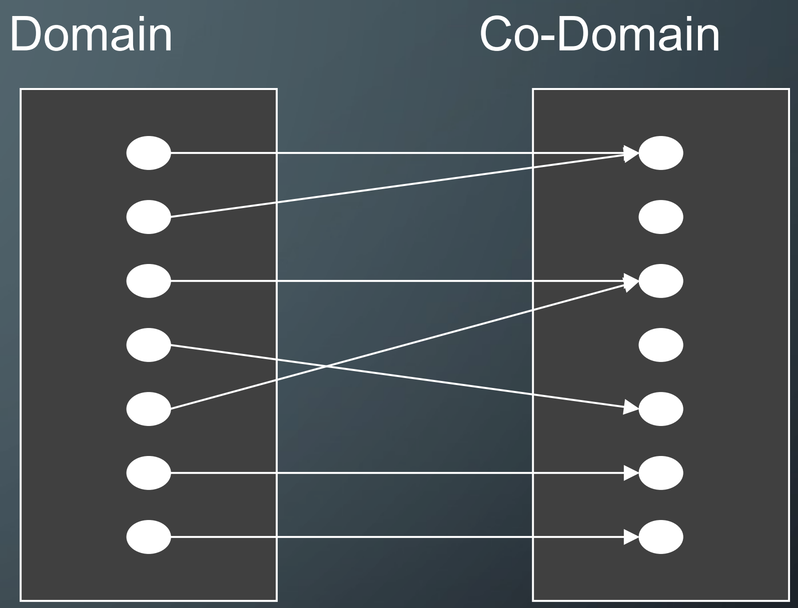
One-to-one (injective)

Different inputs yield different outputs
No two inputs share an output

ONTO, SURJECTIVE FUNCTIONS



SURJECTIVE FUNCTION



NON-SURJECTIVE FUNCTION

Everything in Co-Domain “receives” something

PROPERTIES OF FUNCTIONS

One-to-one (injective)

$$x \neq y \Rightarrow f\left(x\right) \neq f\left(y\right)$$

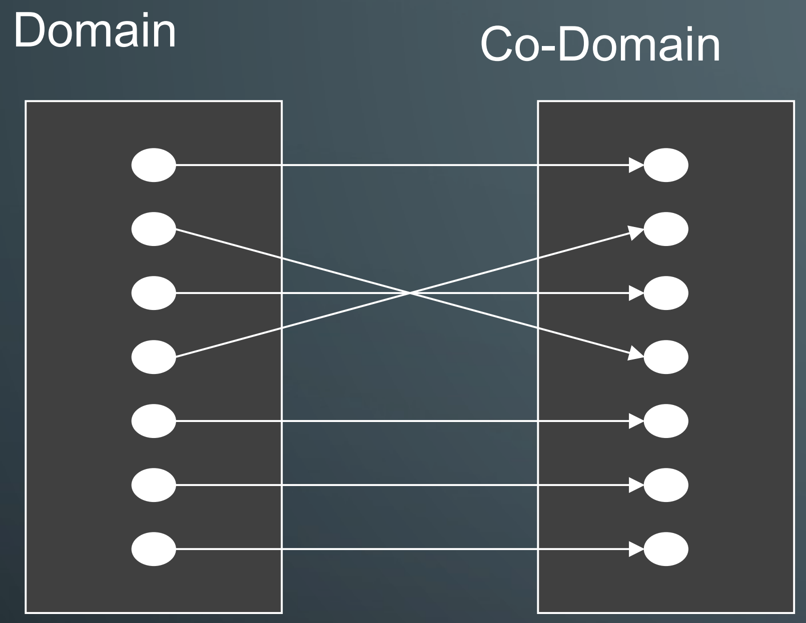
Onto (surjective)

Everything in is the output of something in

PROPERTIES OF FUNCTIONS

- One-to-one (injective)
- Onto (surjective)
- One-to-one Correspondence (bijective)
  - Both one-to-one and surjective
  - Everything in is mapped to by a unique element in
  - All elements from domain and co-domain are perfectly “partnered”

BIJECTIVE FUNCTIONS



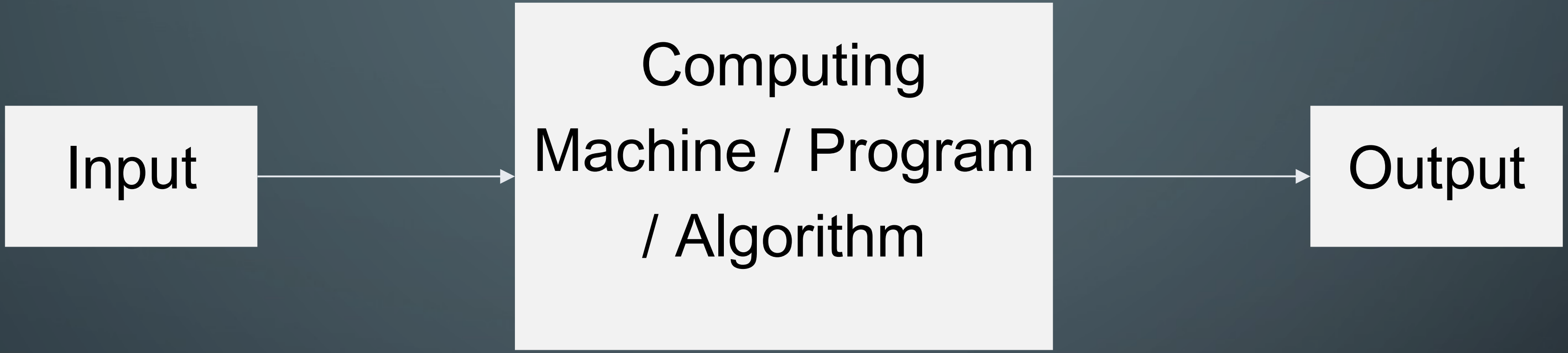
BIJECTIVE FUNCTION

Because Onto:
Everything in Co-Domain “receives” something

Because 1-1:
Nothing in Co-Domain “receives” two things

Conclusion:
Things in the Domain exactly “partner” with things in Co-Domain

OVERVIEW OF COMPUTATION!



Input / Output are ***Strings***

Computing Machine is something that ***implements a set of functions*** that we care about.

PART 2: BIRD'S EYE VIEW OF THEORY OF COMPUTATION?

OVERVIEW OF THEORY OF COMPUTATION

Defining Computation

Input → Computing Machine / Program / Algorithm → Output

Computational Models

Circuits < Finite Automata < Pushdown Automata < Turing Machine = RAM Model

Computational Complexity

Decidability

P, NP, NP-Hard

P-Space, Co-NP, etc

PART 3: BACKGROUND MATH

OBJECTS THAT WE'LL NEED

- Sets
- Functions (already started looking at this)
- Strings (already looked at this briefly)

REVIEW: SETS

SETS

- Notation:
- Order does not matter
- No Duplicates
- Equal when they have the same members

SEQUENCES/ TUPLES

- Notation:
- Order matters
- Duplicates allowed
- Equal when they have the same elements in the same order

REVIEW: SET OPERATORS

- or

REVIEW: SET OPERATORS

- Set equality
- Set membership
- Subset
- Superset
- Proper Subset
- Proper Superset

- or

- Set Union
- Set Intersection
- Set Difference
- Cross Product
- Power Set

IMPORTANT SETS

IMPORTANT SETS

Null Set; Set containing nothing

Natural Numbers = {0, 1, 2, 3, …}

Integers = {…, -3, -2, -1, 0, 1, 2, 3, …}

Rational Numbers =

Set containing 0 and 1 (common alphabet for binary #s)

SET CARDINALITY

- The number of distinct members of a set.
  - How many things I could put on the left hand side of to make the statement true?

Pop Quiz:

SET BUILDER NOTATION

Version 1:

- “The set of all members of that make true”

Version 2:

- “The set of all results of when applied to members of our universe that make true”

COMPARING CARDINALITIES WITH FUNCTIONS

- Two sets have the same cardinality ***if there is a bijection between them***
- What does it mean for a set to have cardinality 5?
  - It has a bijection with the set
- A ***finite set*** has cardinality if it has a bijection with the set
- An ***infinite set*** has no bijections with any set for

ARE ALL FUNCTIONS COMPUTABLE?

- How could we approach this question?

IMPLEMENTING A FUNCTION

- Examples of ways to implement a function:
- Properties we want of implementations:

COMING UP!

- For any “reasonable” model of computing, there will be some uncomputable functions

REVIEW OF THIS DECK!

- A computer is any process that can take strings as input and produce strings as output.
- Computation of that machine / computer is a function (it maps inputs to outputs)
- Moving forward:
  - More math background
  - Different types of computers and the pros / cons of each