Programming logic:
When code does what you say, not what you mean

University of Bologna - 10/07/2025
Lorenzo Dall'Olio, Ph.D.,
Health Researcher for IRCCS ISNB Bellaria

What are we going to learn?

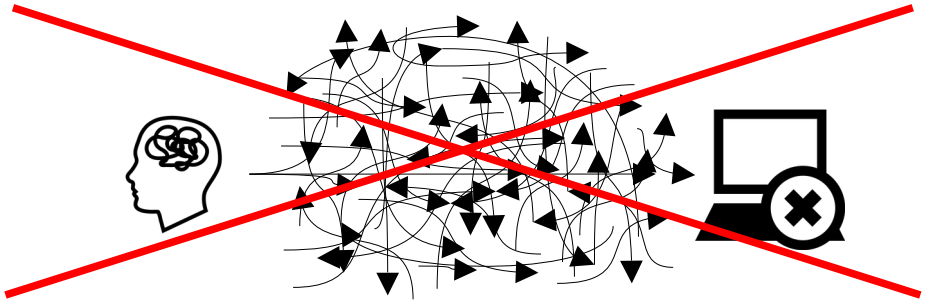
What are we going to learn?

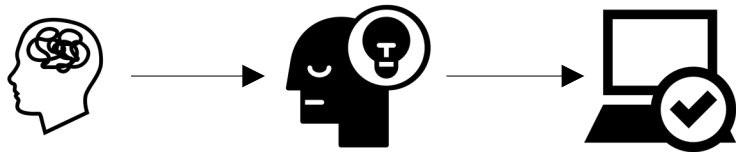
Nothing

What are we going to learn?

~~Nothing~~

To properly craft **the** algorithm we wanted to





How to properly craft an algorithm?

How to properly craft an algorithm?

Let’s learn it by **brute force**.

How to properly craft an algorithm?

Let’s learn it by **brute force…** on yourselves!

The Algorithm Challenge - Rules

- You will write **10 .txt files**, each file will provide pseudo-code (Instructions in a bullet format) to achieve a different task. You are allowed to use *if-else* and *loops* syntaxes if needed.

- deliver the 10 .txt files **in a folder** with the team name, and each file named after the task its addressing.

- 4 teams, semifinals and finals format, each challenge will vert on **5 tasks**.

- During semifinals 2 selected by each team, and 1 by me. During the final, the remaining 5 tasks will be evaluated (the 5 you have not performed during semifinals).

- Possible further challenges might be added as tie-breakers by me, the only supreme judge/divinity of this lesson. [least instruction challenges]

- No violence, no self-harm, human compilers will still perform basic life continuation activites without explict instructions. **Please, no LLMs usage** (if you want to learn).

The Algorithm Challenge - Tasks

1) Take your phone out of your pocket
2) lift a chair (for 5 seconds)
3) Hug someone
4) Perform 1 Burpee (1 push up, then 1 jump + overhead clap)
5) Dance (for 5 seconds)
6) Physically exit the room (from a valid door!)
7) From seated position stand up and sit down again
8) Shuffle a deck of cards
9) Order 3 numbered cards by increasing value
10) Find the red covered card from within the deck of cards

How to properly craft an algorithm?

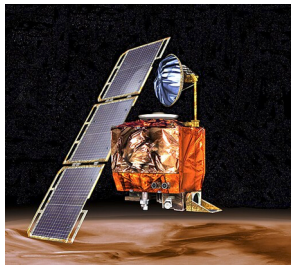
How to properly craft an algorithm?

- **Unambigous**

How to properly craft an algorithm?

- **Unambigous**

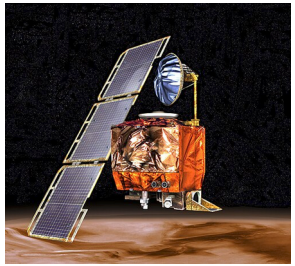
  Mars Climate Orbiter, 1998



How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998

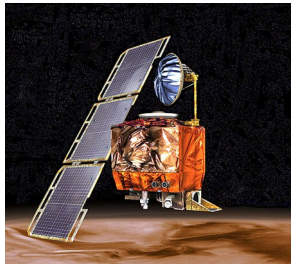


(Newton or Pound-force?)

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998

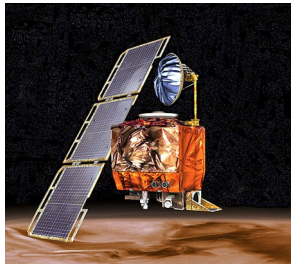


(Newton or Pound-force?) → disintegrated by → mars atmosphere

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



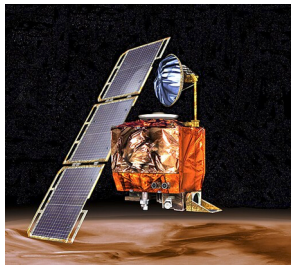
(Newton or Pound-force?) → disintegrated by → mars atmosphere

- **Possible**

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



(Newton or Pound-force?) → disintegrated by → mars atmosphere

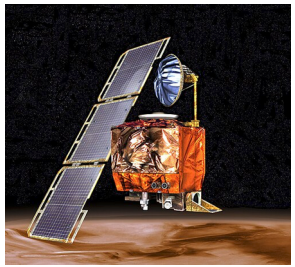
- **Possible**

The USS Yorktown, 1997

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



(Newton or Pound-force?) → disintegrated by → mars atmosphere

- **Possible**

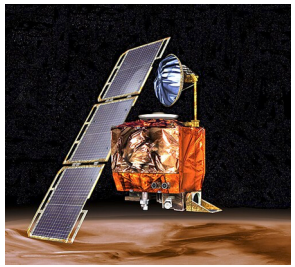
The USS Yorktown, 1997

(divide hp by 0)

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



(Newton or Pound-force?) → disintegrated by → mars atmosphere

- **Possible**

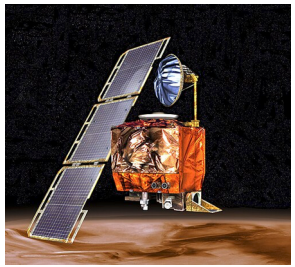
The USS Yorktown, 1997

(divide hp by 0) → stalled for hours → in open sea

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



(Newton or Pound-force?) → disintegrated by → mars atmosphere

- **Possible**

The USS Yorktown, 1997

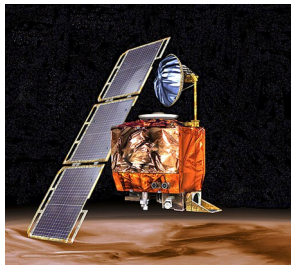
(divide hp by 0) → stalled for hours → in open sea

- **Correct**

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



(Newton or Pound-force?) → disintegrated by → mars atmosphere

- **Possible**

The USS Yorktown, 1997

(divide hp by 0) → stalled for hours → in open sea

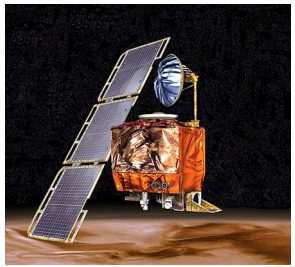
- **Correct**

“Prime numbers are those not divisible by any other number” - You, in 5th grade.

How to properly craft an algorithm?

- **Unambigous**

Mars Climate Orbiter, 1998



(Newton or Pound-force?) → disintegrated by → mars atmosphere

- **Possible**

The USS Yorktown, 1997

(divide hp by 0) → stalled for hours → in open sea

- **Correct**

“Prime numbers are those not divisible by any other number” - You, in 5th grade.

→ There are no → prime numbers

How to properly craft an algorithm?

...**Atomize**...

How to properly craft an algorithm?

...**Atomize**...

...and check for correctness, disambiguate, verify the possibility to yadda yadda yadda…

How to properly craft an algorithm?

...**Atomize**...

...and check for correctness, disambiguate, verify the possibility to yadda yadda yadda…

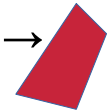
Atomize!!!

How to properly craft an algorithm?

In general:

- Minimize all possible assumptions.

"*Draw a quadrilateral figure*" → it does not state it to be a square, a rectangle, a parallelorgam, a trapezoid, etc… →



- Maximize possible starting scenarios, and connect them to a common case in a funnel fashion.

"*Horizontally flip a sequence of numbers*" → how is this sequence stored? Can it have length of 1 element?

- Decompose as much as possible.

The more, the better, always! Then you can use the prebuilt functions optimized by experts to jump multiple steps all at once.

Let’s make a useful one

Now make a pseudo-code algorithm to perform one of the following tasks:

- Find all prime numbers between 1 and N **[Very Easy]**

- Swap 2 numerical variables values without using a third variable or a pre-built function **[medium]**

- Solve the Hanoi tower with N starting disks **[Very Hard]**
check https://www.mathsisfun.com/games/towerofhanoi.html for visual help

Let’s make a useful one – solution 1

Find all prime numbers between 1 and N **[Very Easy]**:
{Sieve of Eratosthenes}

- Store all numbers from 2 to N in increasing order
  i=2, Repeat until i > $N^{0.5}$
- delete all multiples of i from the list
- assign the next smallest value to i

Let’s make a useful one – solution 2

Swap 2 numerical variables value without using a third variable or a pre-built function **[medium]**

- A = A+B (store the sum in the first variable)
- B = A-B (store the sum minus the second variable in the second v.)
- A = A-B (store the sum minus the second variable in the first var.)

Let’s make a useful one – solution 3

Solve the Hanoi tower with N starting disks **[Very Hard]**:

- Assign each disk a number, 1 for the smallest, N to the largest
  i=0, Repeat 2^N-1 times (until all disks are in tower 3)
- A = bitwise XOR between i and i+1 binary representations
- P = the largest position (power of 2) of A having a 1 instead of a 0
  If P%2 is equal to N%2 (P and N have the same parity)
- Move disk P forward of 1 tower in modulus 3 (tower 3 → tower 1)
  else
- Move disk P back of 1 tower in modulus 3 (tower 1 → tower 3)
- i = i+1