Graphical Loop Invariant Programming in CS1

Simon Liénardy, Lev Malcev, Benoit Donnet

simon.lienardy@uliege.be

Agenda

- Context
- Programming Methodology
- GLI
- CAFÉ
- Preliminary Evaluation
- Conclusion & Future Work

Context: First Year Students in CS

- Introduction to programming course
  - University of Liège, Belgium
  - Open access to the University (and Higher Ed. in general)
    - No background required, esp. in Mathematics
- ~ 80 Students in Computer Science (Bloc 1)
- Programming skills required by next courses

Programming Methodology

```
INIT
{Invariant}
while(B){
  {Invariant ∧ B}
   LOOP BODY
  {Invariant}
}
{Invariant ∧ ¬ B}
END
```

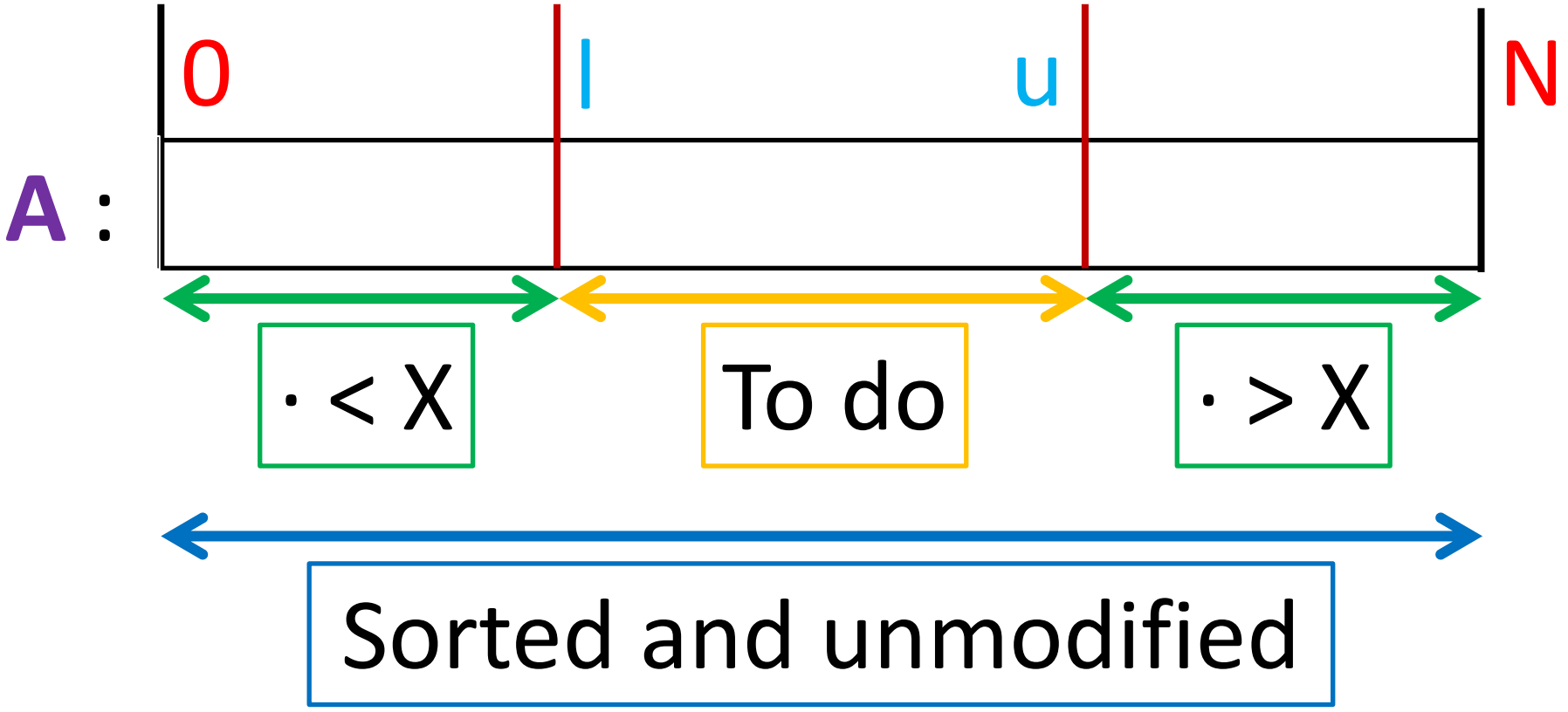
Deriving the code:

- Based on Dijkstra, *A Discipline of Programming* (1976)
- Graphical representation
- Represent what has already been computed

⇒ Strategy to solve the problem (*Metacognition*)

Graphical Loop Invariant: example Binary search in a sorted Array

Value searched : X



Graphical Loop Invariant: Guidelines

1. Drawing must be relevant and named
2. The boundaries of the problem are provided
3. There must be one (or more) dividing line(s)
4. Each dividing line should be labeled (w/ variables)
5. Label(s) about what has been achieved so far
6. Label(s) about what should be done
7. All the named elements and variables are present in the code

Graphical Loop Invariant: Rules

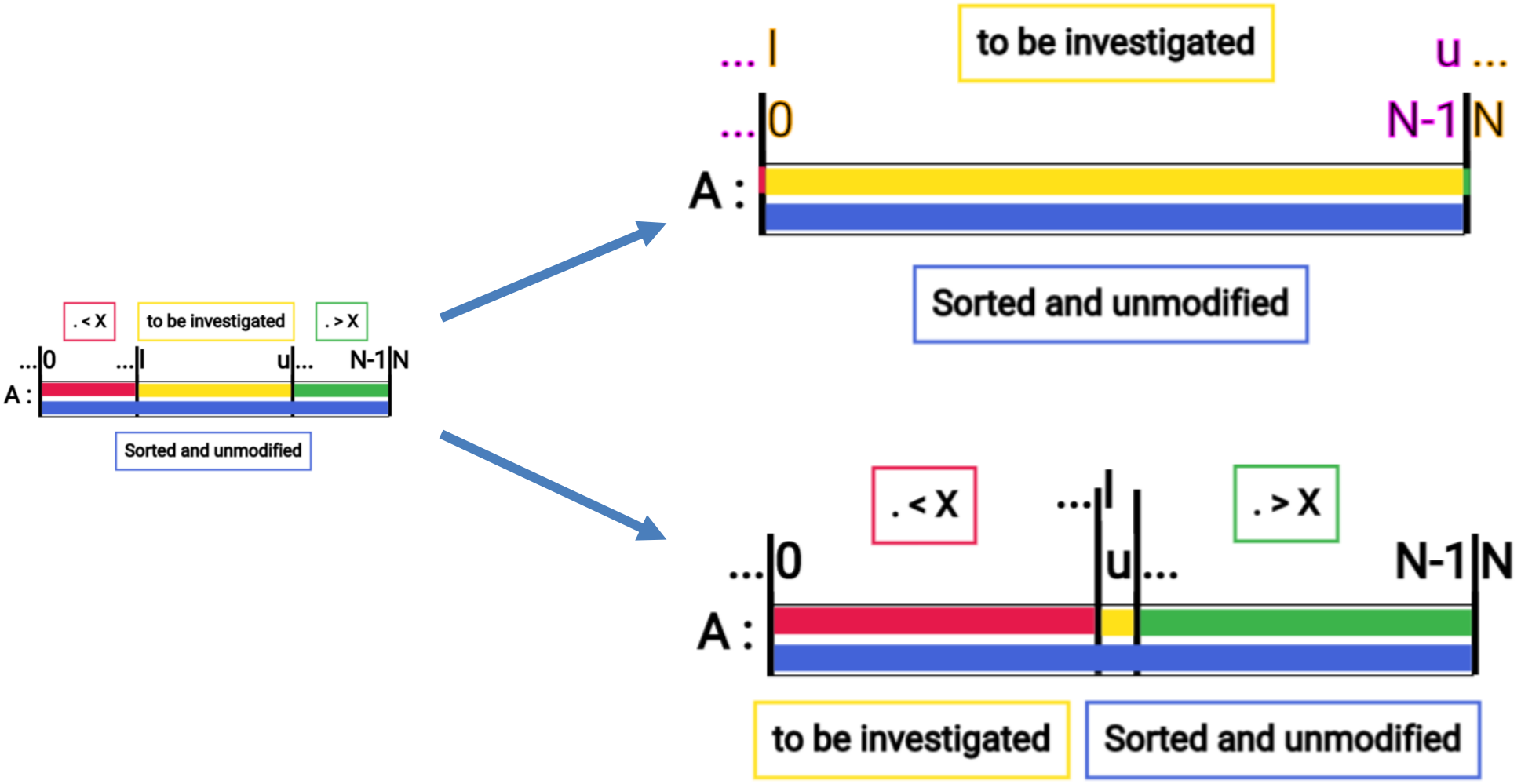
- Rules can be used to assess a Student's Invariant
- Mistakes can be sorted into 3 categories:
  - Syntax

    E.g. missing elements
  - Semantic

    E.g. labels that do not make sense or not relevant w/ the problem
  - Matching with the code

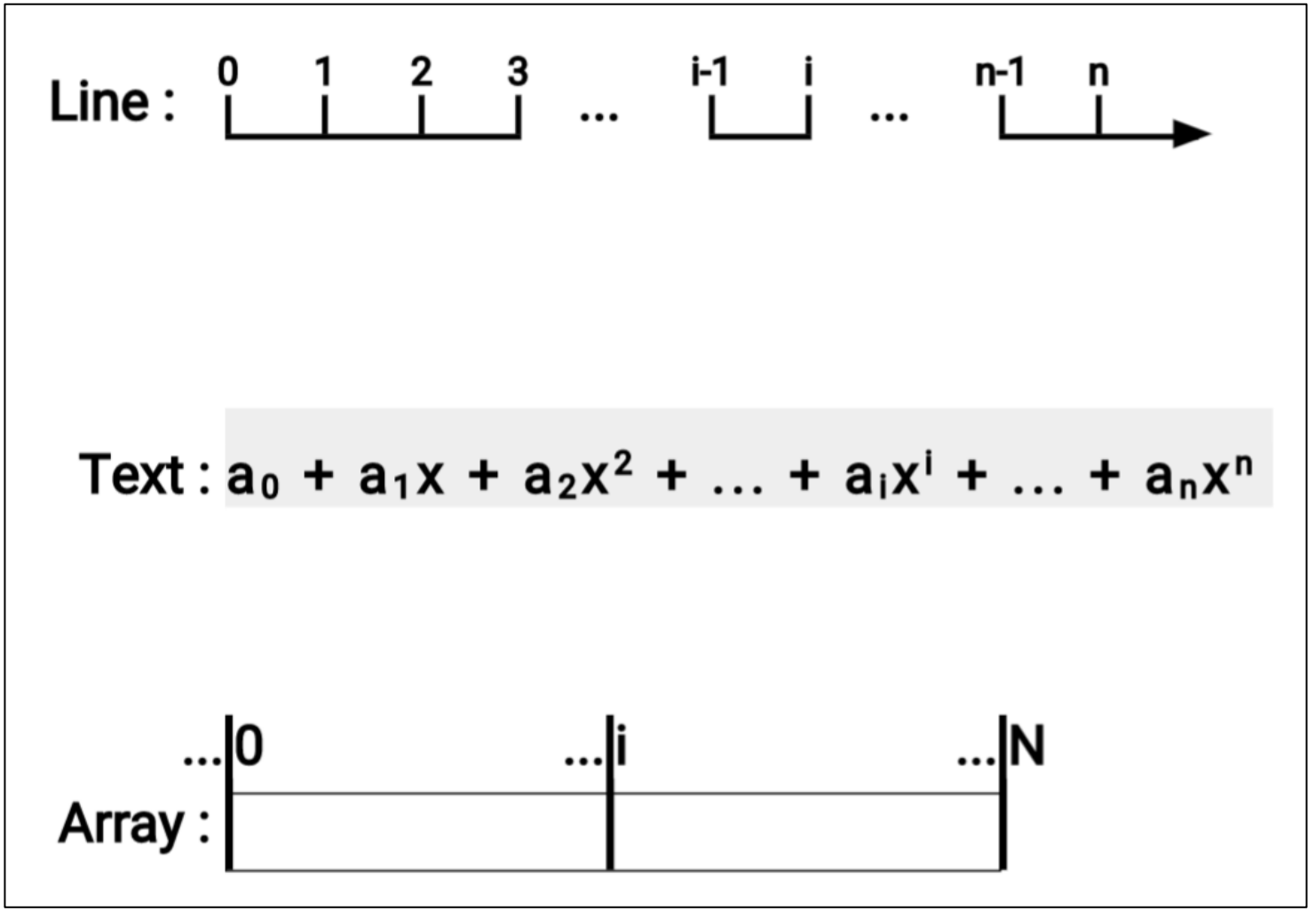
    The Invariant should be used to write the code

Deriving the code from the Invariant

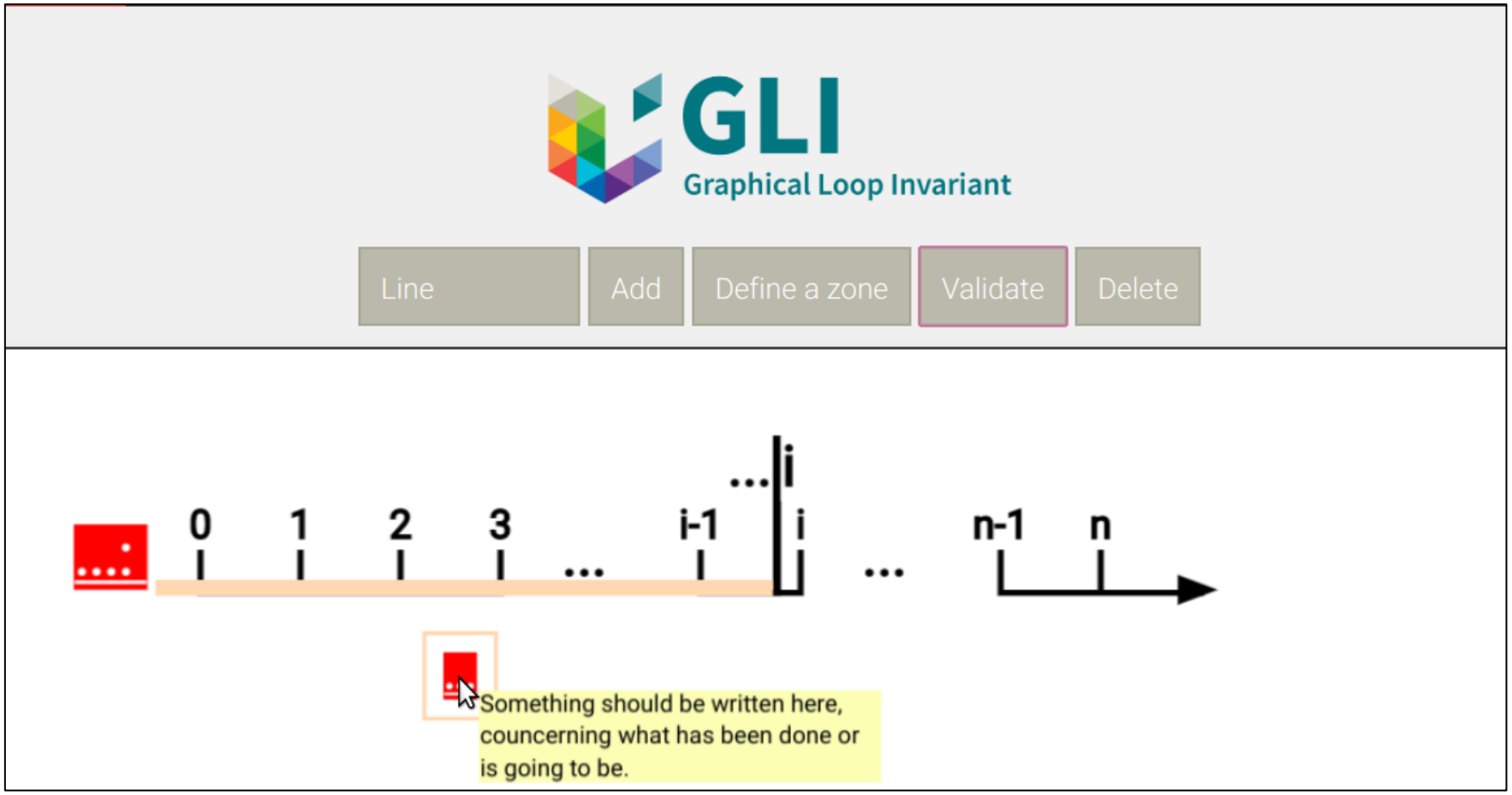


Introducing GLI

GLI: Patterns available



GLI: Syntax checks



GLI: Deriving the code

As soon as the checks are passed:

- One can move the dividing bars to get the init state:


```
int l = 0;
int u = N-1;
```

- Or move them to find a condition under which the loop must be stopped
  → Infer the loop condition

Introducing CAFÉ

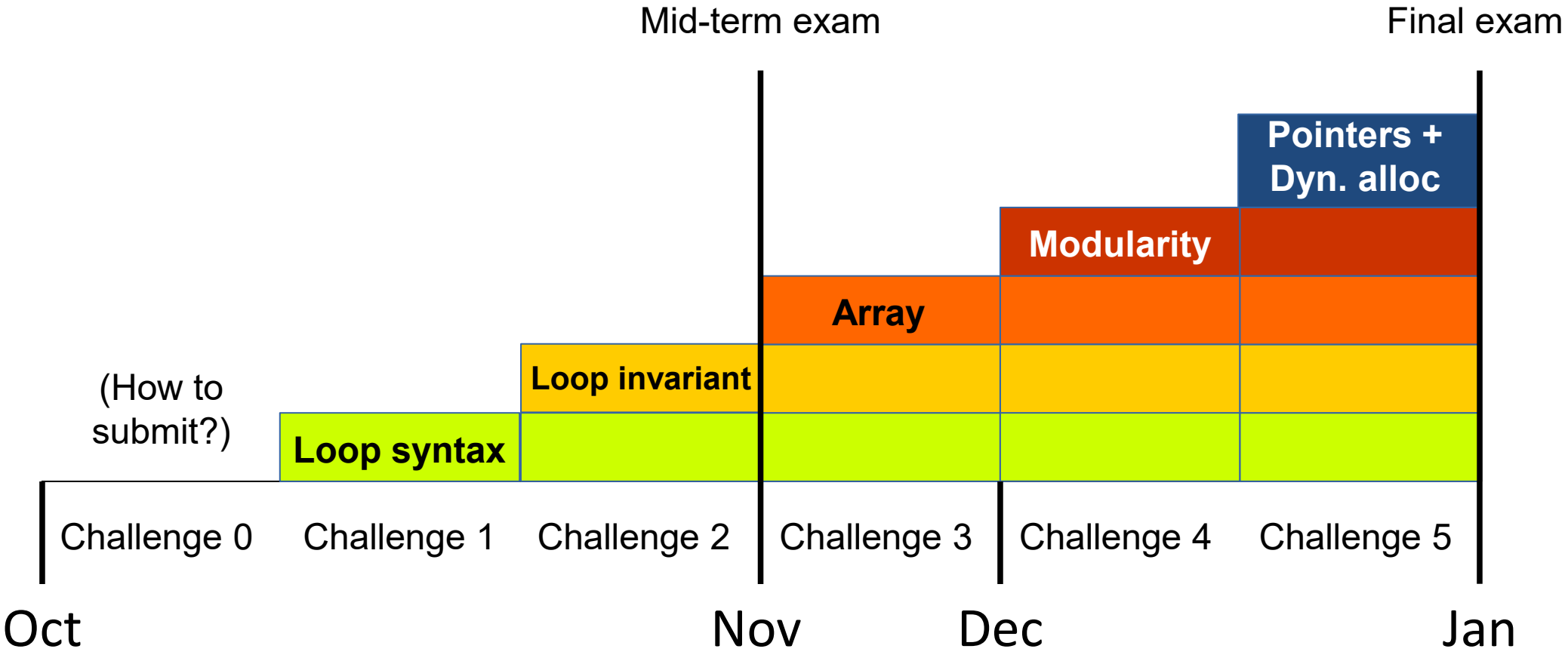
- French acronym for

  Correction Automatique et Feedback des Étudiants

- Students submit on a web platform 5 Challenges of increasing difficulty during the semester
  - Plus Challenge 0 to learn how to submit
- Assessment for Learning oriented [Sambel et al., 2013, Wiliam, 2011]

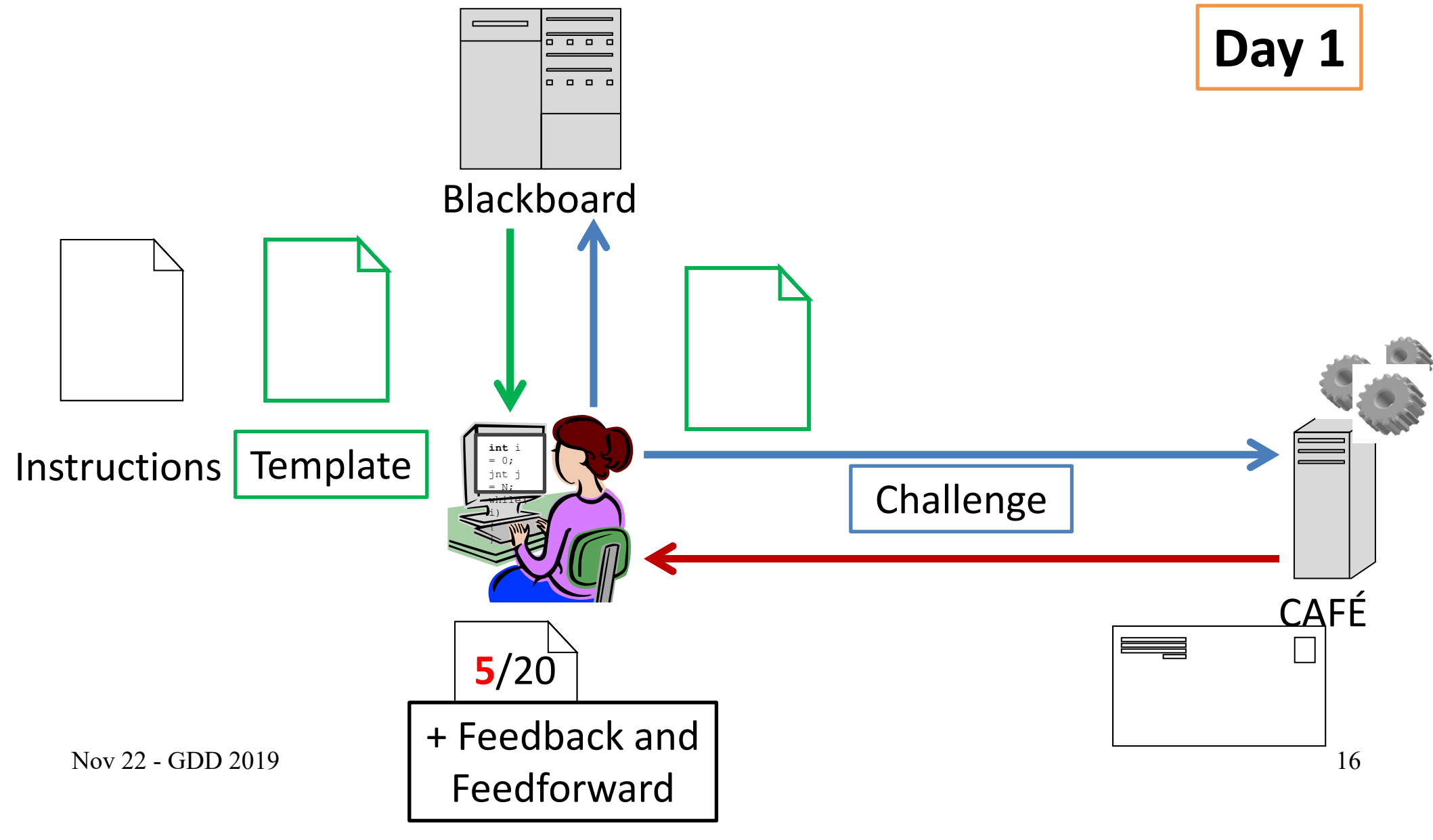
S. Liénardy, L. Leduc D. Verpoorten and B. Donnet. 2020. Café: Automatic Correction and Feedback of Programming Challenges for a CS1 Course. In *Proc. ACE '20: Twenty-Second Australasian Computing Education Conference*

Challenges?

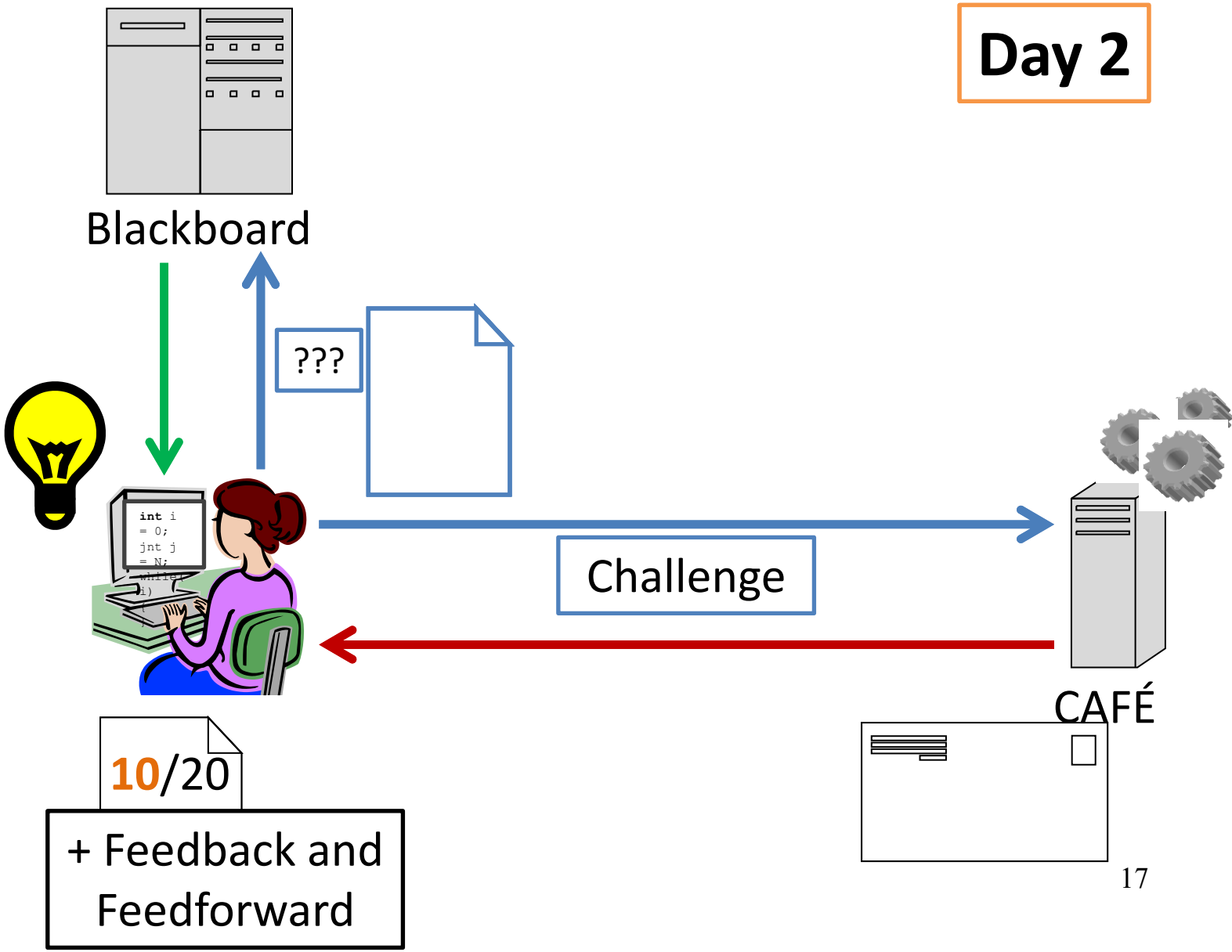


→ Cumulative difficulty

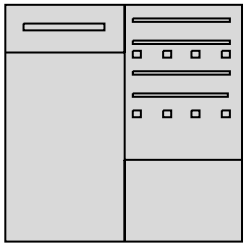
Introducing CAFÉ



Introducing CAFÉ



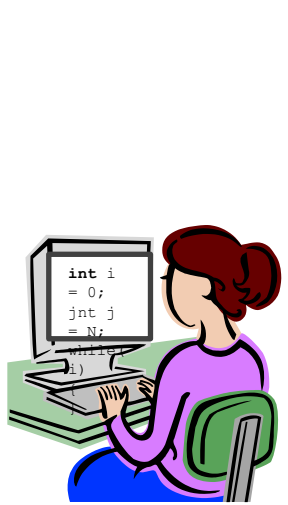
Introducing CAFÉ



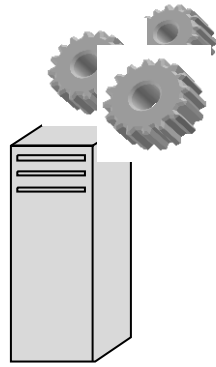
Blackboard

Day 3

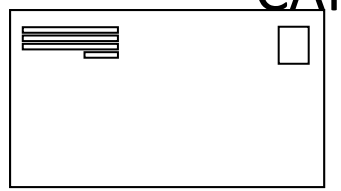
→ Closing the feedback loop [Boud, 2000]



Challenge



CAFÉ



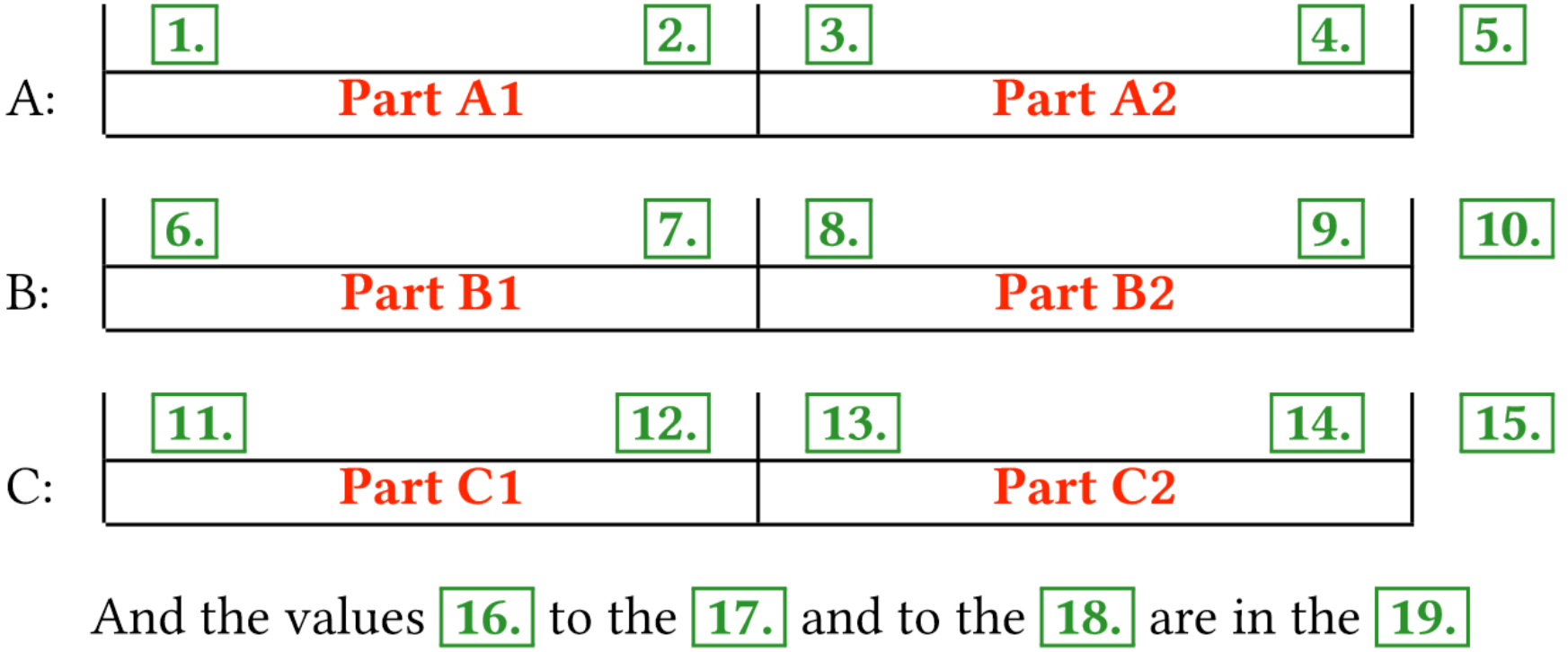
19/20

+ Feedback and Feedforward

CAFÉ: Submitting Invariant

- Graphical Loop Invariant is the **corner stone** of our methodology
- How to make students work with the Invariant during the Challenges?
- Blank Invariant
  - To be filled by the student
  - Bootstrap effect

CAFÉ: Submitting Invariant (Challenge : compute $C = A \cap B$)



1. -> **15.** : Replace by variables, constants, numerical values

16. : Replace by “different from”, “common to “,etc.

17. -> **19.** : Replace by **Part A1**, **Part A2**, **Part B1**,…

CAFÉ: Invariant checking

- Variables present in the Inv are in the code too
  - And initialized according to the Invariant
- Array Indices in the Inv are used to index arrays
  - Out of Bound check
- Loop Variant correction
- Iterations count (if complexity constrains)
- Unit tests
- Feedback & Feedforward added after correction

CAFÉ: Preliminary Evaluation

- Data about :
  - Students participation
  - Performance
  - Perception
- Over multiple years
- Work in progress

Conclusion

- GLI:
  - Evaluation = work in progress
- CAFÉ:
  - Takes time but efficient and scalable
  - Language independent (modulo slight modifs)
  - Lot of data to be analyzed
- Questions ?