ALGORITHM AND COMPUTATIONAL THINKING 2

WEEK 1 – KICK OFF !

CADT
IDT

KICK OFF OBJECTIVES

Kick Off

- Term 2 **agenda** & **learning objectives**
- Term 2 expected **habits** & **attitudes**
- Term 2 **evaluation**

Let's start

- **Term 1 refresher**
- **Strategies** to solve problems with arrays

Agenda - Term 2

Week	Objectives	EXAMS	PROJECTS
W 01	Arrays Problem Solving		
W 02	Functions & Modular Design (Part 1)		
W 03	Functions & Modular Design (Part 2)		
W 04	2D Arrays Problem Solving		
W 05	Revisions & Mid Term exam	MID TERM	
W 06	Pointers		PROPOSAL
W 07	Strings & Pointer Manipulation		
W 08	Structures & Data Modeling		
W 09	Dynamic Memory Allocation		
W 10	Revisions & Project Jury		JURY
W 11	Final Exam	FINAL	

Note: This agenda is subject to change.

Practical skills you need to **acquire** - Term 2

Computational Thinking

- ✓ **Break down complex problems** using functions and design reusable code blocks.
- ✓ **Debug** programs using **structured thinking** and step-by-step analysis.

Coding

- ✓ Write **structured C code** using arrays, functions, pointers, strings, and structs.
- ✓ Manipulate **strings**.
- ✓ **Manage memory manually** using dynamic allocation (malloc, calloc, free).

Project

- ✓ **Build a complete project** with applied concepts.

Attitudes you need to **adopt** - Term 2

Think by yourself

- ✓ **Trust** yourself !
- ✓ Relying on **ChatGPT** (or any AI) for answers **won't help** your **brain grow.**

Work with others

- ✓ **Compare** your solutions with other students.
- ✓ **Review** other peer works.

Be honest

- ✓ **Don’t lie** if you've asked another student or AI for the solution.
- ✓ Instead, **talk to your teacher** about how you can **improve your behavior**.

Explore

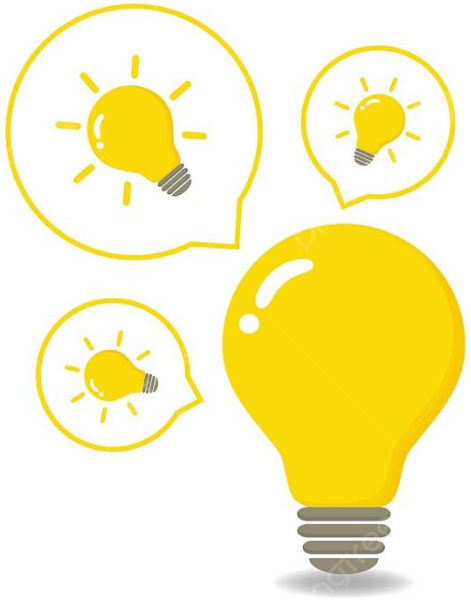
- ✓ Approach tasks with a **spirit of exploration**.
- ✓ **Mistakes** are part of the learning process.

Create

- ✓ Don’t just look for the “right” answer.
- ✓ **Think outside the box** to find your unique solutions.

Good to Know !

The **process** is more important than the **results**.



If you take care of the **process**, you will get the **results**.

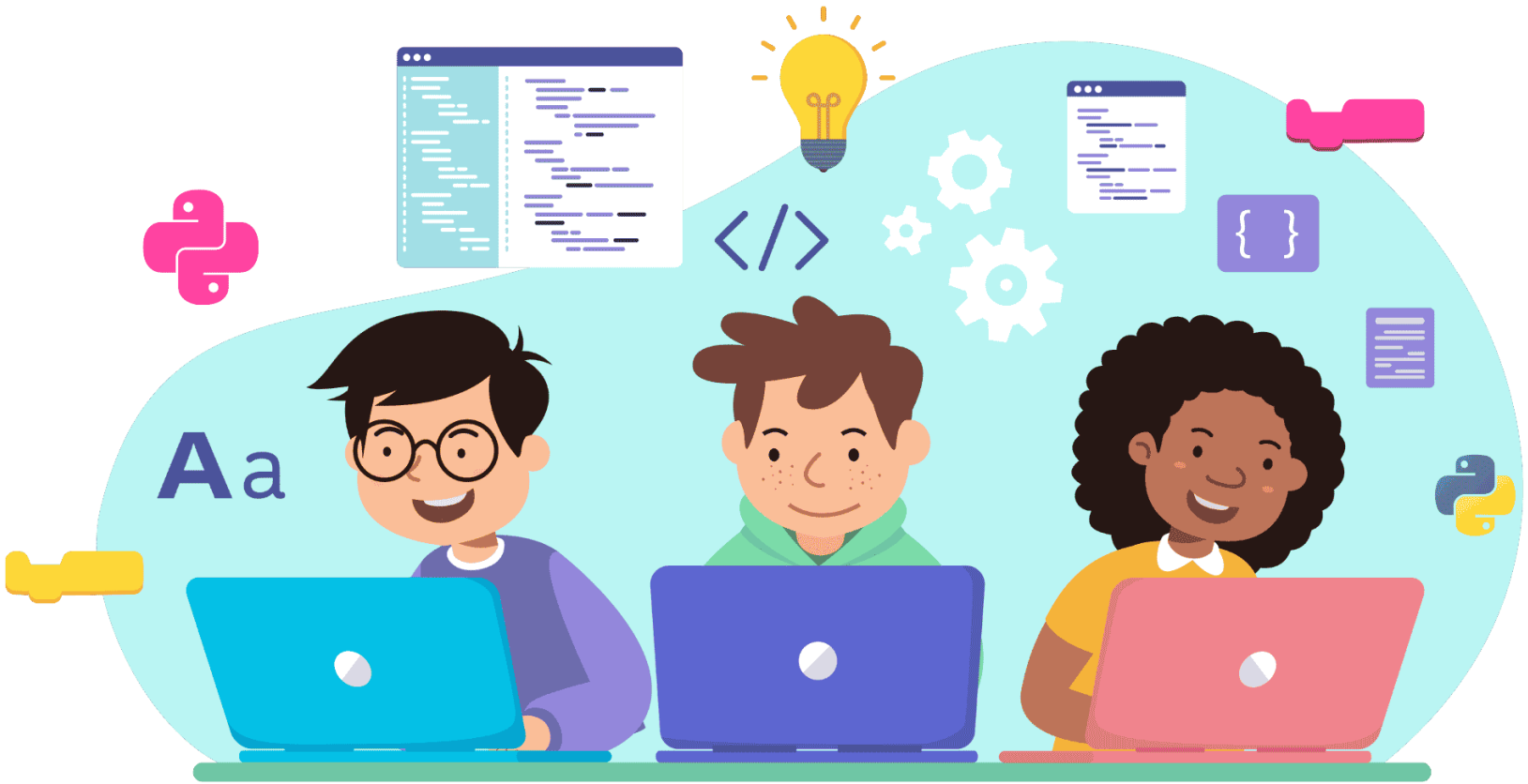
Evaluation - Term 2

TYPE OF EVALUATION	RATIO
Attendance & Participation *-20 points for each missed session*	05 %
Assignment *-20 points for each missed session*	10 %
Quizzes	10 %
Mid Term Exam	15 %
Final Exam	30 %
Project	30 %

Note: This evalaution is subject to change.

ALGORITHM AND COMPUTATIONAL THINKING 2

WEEK 1 – Array Problem Solving



CADT IDT

TERM 1 - **REFRESHER**

QUIZ

Q1 What will this program print?

```
int array[5] = {5, 7, 3, 8, 1};
bool isFound = false;

for (int i = 1; i < 5; i++) {
  if (array[i - 1] == array[i]) {
        isFound = true;
        break;
   }
}

printf("%d\n", isFound);
```

A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

Q1

ANSWER

What will this program print?

```
int array[5] = {5, 7, 3, 8, 1};
bool isFound = false;

for (int i = 1; i < 5; i++) {
  if (array[i - 1] == array[i]) {
        isFound = true;
        break;
   }
}

printf("%d\n", isFound);
```

A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

QUIZ

Q2

What will this program print?

```
int array[5] = {5, 7, 3, 3, 1};
bool isFound = false;

for (int i = 1; i < 5; i++) {
  if (array[i - 1] == array[i]) {
        isFound = true;
        break;
   }
}

printf("%d\n", isFound);
```

A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

Q2

ANSWER

What will this program print?

```
int array[5] = {5, 7, 3, 3, 1};
bool isFound = false;

for (int i = 1; i < 5; i++) {
  if (array[i - 1] == array[i]) {
        isFound = true;
        break;
   }
}

printf("%d\n", isFound);
```

A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

QUIZ

Q3

What will this program print?

```
int array[5] = {1, 2, 3, 4, 5};
bool greater = true;

for (int i = 0; i < 5; i++) {
  if (array[i - 1] > array[i]) {
        greater = false;
        break;
   }
}

printf("%d\n", greater);
```

A) 0

B) 1

C) Compilation Error

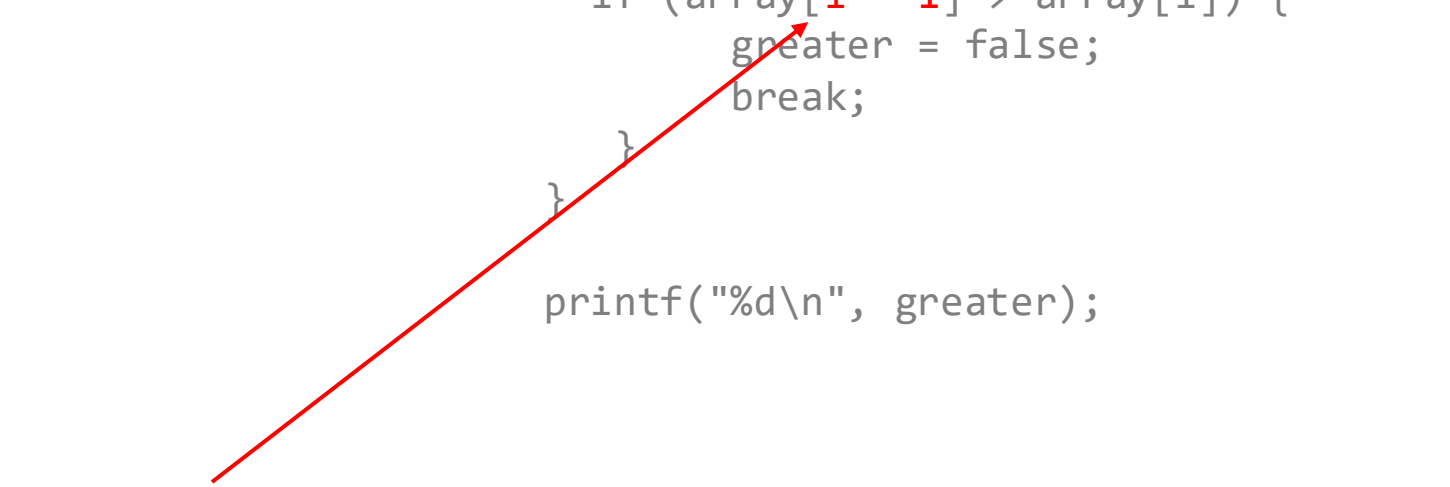
D) Undefined Behavior

Q3

ANSWER

What will this program print?

```
int array[5] = {1, 2, 3, 4, 5};
bool greater = true;

for (int i = 0; i < 5; i++) {
  if (array[i - 1] > array[i]) {
        greater = false;
        break;
    }
}

printf("%d\n", greater);
```



A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

QUIZ

Q4

What will this program print?

```
int array[5] = {5, 7, 5, 8, 8};
bool isFound = false;

for (int i = 0; i < 3; i++) {
  if (array[i + 2] == array[i]) {
        isFound = true;
        break;
   }
}

printf("%d\n", isFound);
```

A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

Q4

ANSWER

What will this program print?

```
int array[5] = {5, 7, 5, 8, 8};
bool isFound = false;

for (int i = 0; i < 3; i++) {
  if (array[i + 2] == array[i]) {
        isFound = true;
        break;
   }
}

printf("%d\n", isFound);
```

A) 0

B) 1

C) Compilation Error

D) Undefined Behavior

QUIZ

Q5

What will this program print?

```
int array[8] = {5, 7, 8, 5, 8, 3, 1, 3};
int value = -1;

for (int i = 2; i < 8; i++) {
  if (array[i] == array[i-2]) {
        value = array[i-2];
   }
}

printf("%d\n", value);
```

A) -1

B) 5

C) 8

D) 3

Q5

ANSWER

What will this program print?

```
int array[8] = {5, 7, 8, 5, 8, 3, 1, 3};
int value = -1;

for (int i = 2; i < 8; i++) {
  if (array[i] == array[i-2]) {
        value = array[i-2];
   }
}

printf("%d\n", value);
```

A) -1

B) 5

C) 8

D) 3

QUIZ

Q6

What will this program print?

```
int array[8] = {5, 7, 8, 5, 8, 3, 1, 3};
int value = -1;

for (int i = 2; i < 8; i++) {
  if (array[i] == array[i-2]) {
        value = array[i-2];
        break;
   }
}

printf("%d\n", value);
```

A) -1

B) 5

C) 8

D) 3

Q6

ANSWER

What will this program print?

```
int array[8] = {5, 7, 8, 5, 8, 3, 1, 3};
int value = -1;

for (int i = 2; i < 8; i++) {
  if (array[i] == array[i-2]) {
        value = array[i-2];
        break;
   }
}

printf("%d\n", value);
```

A) -1

B) 5

C) 8

D) 3

QUIZ

Q7

Which of those array initialization are **correct**?

A.
```
int numbers[5];
```

B.
```
int numbers[];
```

C.
```
int numbers[]  = {4, 3};
```

D.
```
int numbers[3] = {5, 2};
```

Q7

ANSWER

Which of those array initialization are **correct**?

A. `int numbers[5];`

Declares an array of 5 integers with **uninitialized** values (undefined contents)

B. `int numbers[];`

C requires the array size to be **explicitly stated** unless an initializer is provided

C. `int numbers[] = {4, 3};`

The size is inferred to be 2 from the initializer list, **if you're initializing right away**

D. `int numbers[3] = {5, 2};`

This initializes the first two elements , and the last one is set to 0

Strategies to solve problems with arrays

USE
PREVIOUS/NEXT ELEMENTS

CHALLENGE 1

CHECK SOMETHING **ON EVERY ELEMENTS**

CHALLENGE 2

Check something **on at least one element**

CHALLENGE 3

Break
an array
loop

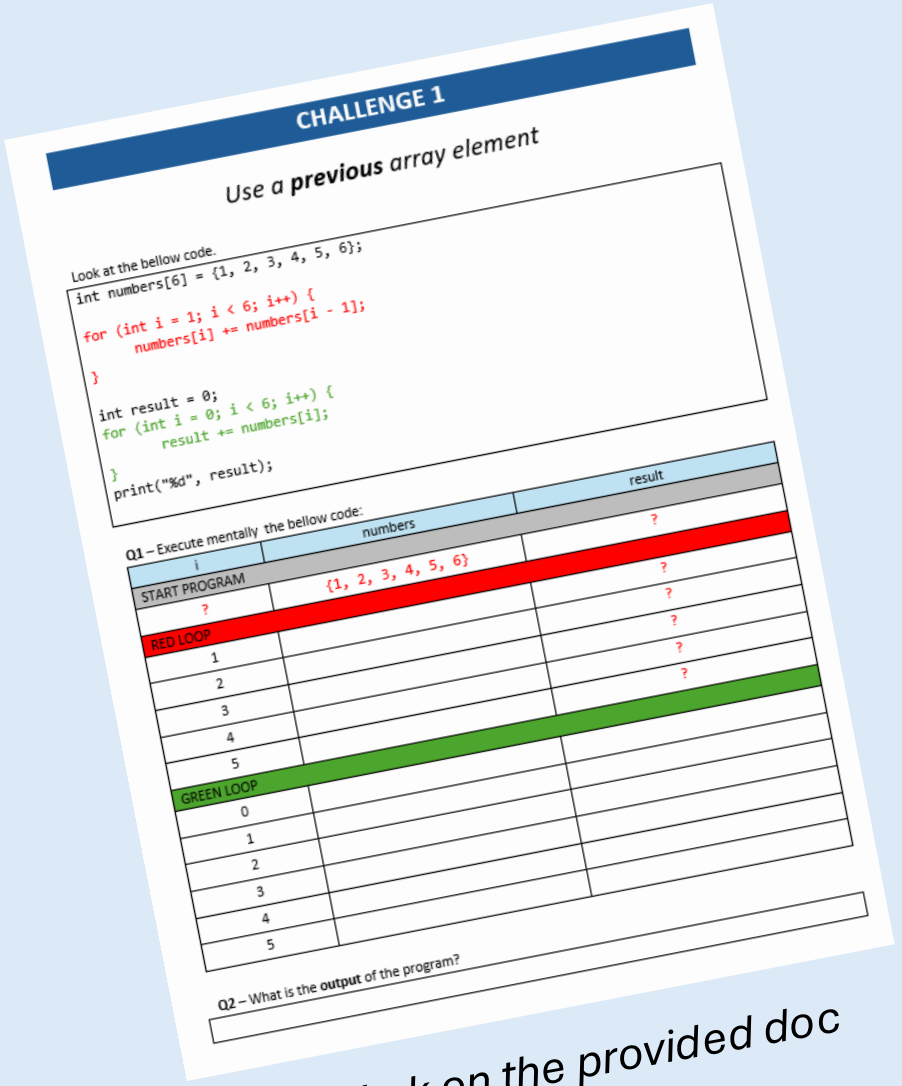
CHALLENGE

USE

PREVIOUS/NEXT ELEMENTS

CHALLENGE 1

10 MIN



CHALLENGE 1

Use a **previous** array element

Look at the bellow code.

```
int numbers[6] = {1, 2, 3, 4, 5, 6};
for (int i = 1; i < 6; i++) {
    numbers[i] += numbers[i - 1];
}
int result = 0;
for (int i = 0; i < 6; i++) {
    result += numbers[i];
}
print("%d", result);
```

Q1 – Execute mentally the bellow code:

i	numbers	result
START PROGRAM	{1, 2, 3, 4, 5, 6}	?
?		?
RED LOOP		?
1		?
2		?
3		?
4		?
5		
GREEN LOOP		
0		
1		
2		
3		
4		
5		

Q2 – What is the **output** of the program?

Work on the provided doc

CHALLENGE 1 CODE EXECUTION

numbers[i-1] is the previous element of numbers[i]

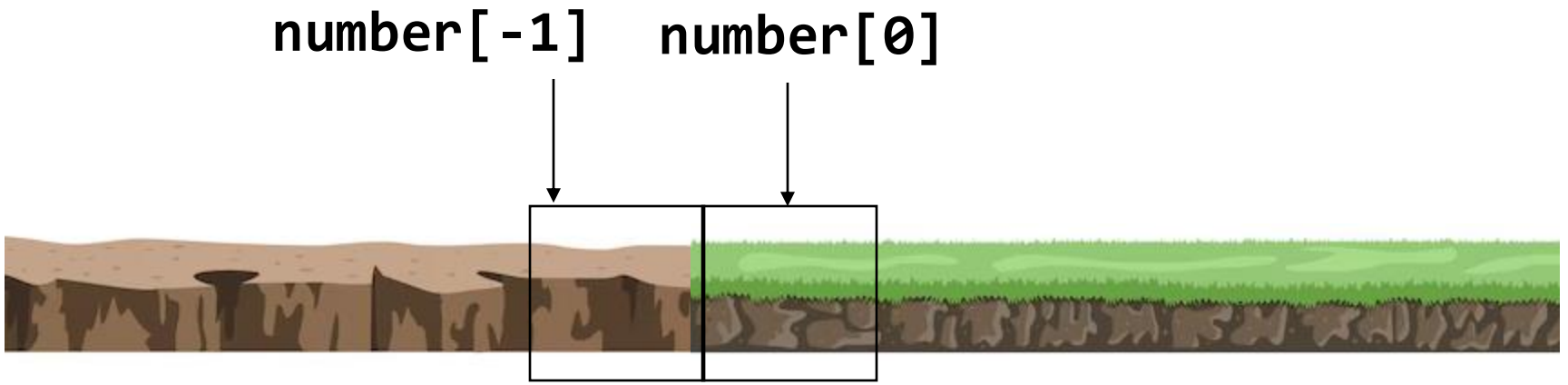
```
int numbers[6] = {1, 2, 3, 4, 5, 6};

for (int i = 1; i < 6; i++) {
        numbers[i] += numbers[i - 1];
}


int result = 0;


for (int i = 0; i < 6; i++) {
        result += numbers[i];
}

print("%d", result);
```

i	numbers	result
START PROGRAM		
?	{1, 2, 3, 4, 5, 6}	?
RED LOOP		
1	{1, 3, 3, 4, 5, 6}	?
2	{1, 3, 6, 4, 5, 6}	?
3	{1, 3, 6, 10, 5, 6}	?
4	{1, 3, 6, 10, 15, 6}	?
5	{1, 3, 6, 10, 15, 21}	?
?	{1, 3, 6, 10, 15, 21}	0
GREEN LOOP		
0	{1, 3, 6, 10, 15, 21}	1
1	{1, 3, 6, 10, 15, 21}	4
2	{1, 3, 6, 10, 15, 21}	10
3	{1, 3, 6, 10, 15, 21}	20
4	{1, 3, 6, 10, 15, 21}	35
5	{1, 3, 6, 10, 15, 21}	56

CHALLENGE 1 QUIZ

What's happened here ?

```
int numbers[6] = {1, 2, 3, 4, 5, 6};
for (int i = 0; i < 6; i++) {
     numbers[i] += numbers[i - 1];
}
```

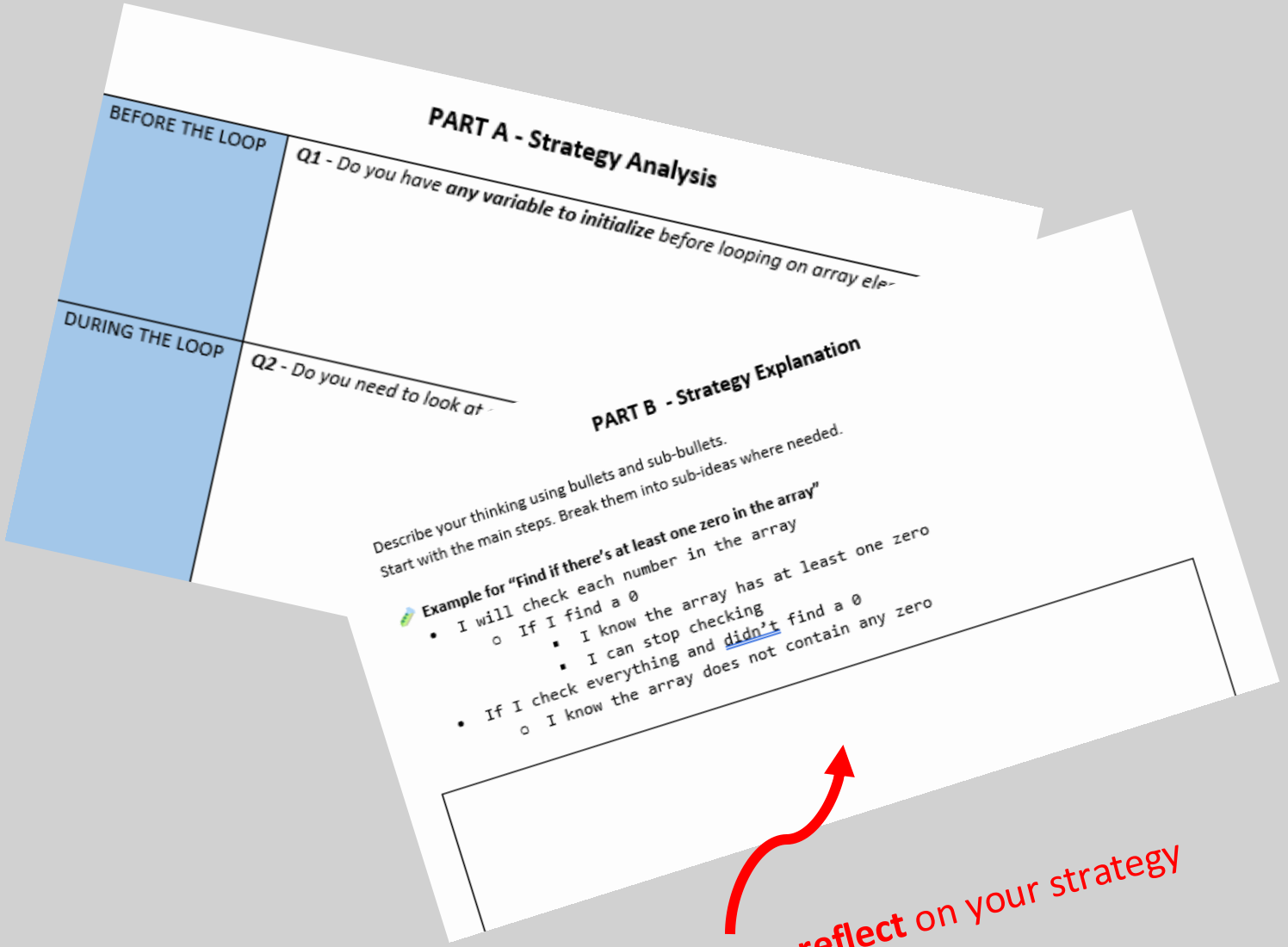


During the first iteration, I is equal to 0 ➔ number[I - 1] will refer to a non allocated area

CHECK SOMETHING ON EVERY ELEMENTS

CHALLENGE 2

10 MIN



You don't **need to code**, but to **reflect** on your strategy

CHALLENGE 2 STRATEGY ANALYSIS

"We want to check that an array is composed only of numbers 3"

BEFORE THE LOOP	***Q1*** *- Do you have* ***any variable to initialize*** *before looping on array elements?* We initialize our variable (only_3) to true.
DURING THE LOOP	***Q2*** *- Do you need to look at* ***every element****, or* ***just some****?* We need to loop on every elements. ***Q3*** *- What are you* ***comparing or counting****?* We are checking if each element is equal to 3 or not. ***Q4*** *- What* ***condition would break*** *the rule or stop your check early?* If 1 element is NOT equal to 3, our variable (only_3) is set to false. Then, we can break our loop.
AFTER THE LOOP	***Q5*** *- What do you need to do after the loop?* We print the result depending on our variable (only_3)

CHALLENGE 2 STRATEGY PLAN

Describe your thinking using **bullets and sub-bullets**.

Start with the main steps. Break them into sub-ideas where needed.

- We define a boolean variable (only_3)
- We set only_3 to true

- We go through the array elements
  - For each number, we check if it's a 3
  - If not, we set only_3 to false

- We print the result depending on only_3

CHALLENGE 2 QUIZ

Which pseudo is faster for challenge 2?

```
INPUT array
SET only3 TO True


FOR index FROM 0 TO 4
     IF array[index] != 3
           only3 = False



PRINT only3?  "only 3" : "not only 3"
```

A

```
INPUT array
SET only3 TO True

SET index TO 0
WHILE index <5 AND only3
    IF array[index] != 3
          only3 = False
    index++


PRINT only3?  "only 3" : "not only 3"
```

B

CHALLENGE 2 QUIZ

ANSWER

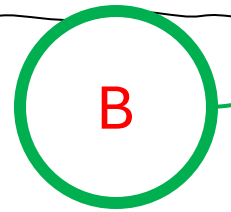
Which pseudo is faster for challenge 2?

```
INPUT array
SET only3 TO True


FOR index FROM 0 TO 4
     IF array[index] != 3
           only3 = False



PRINT only3?  "only 3" : "not only 3"
```

A

Even if the condition is false already
We need to continue till the end

```
INPUT array
SET only3 TO True

SET index TO 0
WHILE index <5 AND only3
    IF array[index] != 3
          only3 = False
    index++



PRINT only3?  "only 3" : "not only 3"
```



We are **breaking the loop** if we know the condition
Is already false

CHALLENGE 2 QUIZ

Will this code work?

```
INPUT array
SET only3 TO True

FOR index FROM 0 TO 4
     IF array[index] != 3
            only3 = False
      ELSE
            only3 = True


PRINT only3?  "only 3" : "not only 3"
```

A Yes, it will work.

B No, it will not work.

CHALLENGE 2 QUIZ

Will this code work?

ANSWER

```
INPUT array
SET only3 TO True

FOR index FROM 0 TO 4
     IF array[index] != 3
           only3 = False
      ELSE
           only3 = True


PRINT only3?  "only 3" : "not only 3"
```

Example of failure :
{3, 3, 3, 1, 3}

Iteration 4:
Element = 1 = > only3 = False

Iteration 5
Element=3 => only3 = True

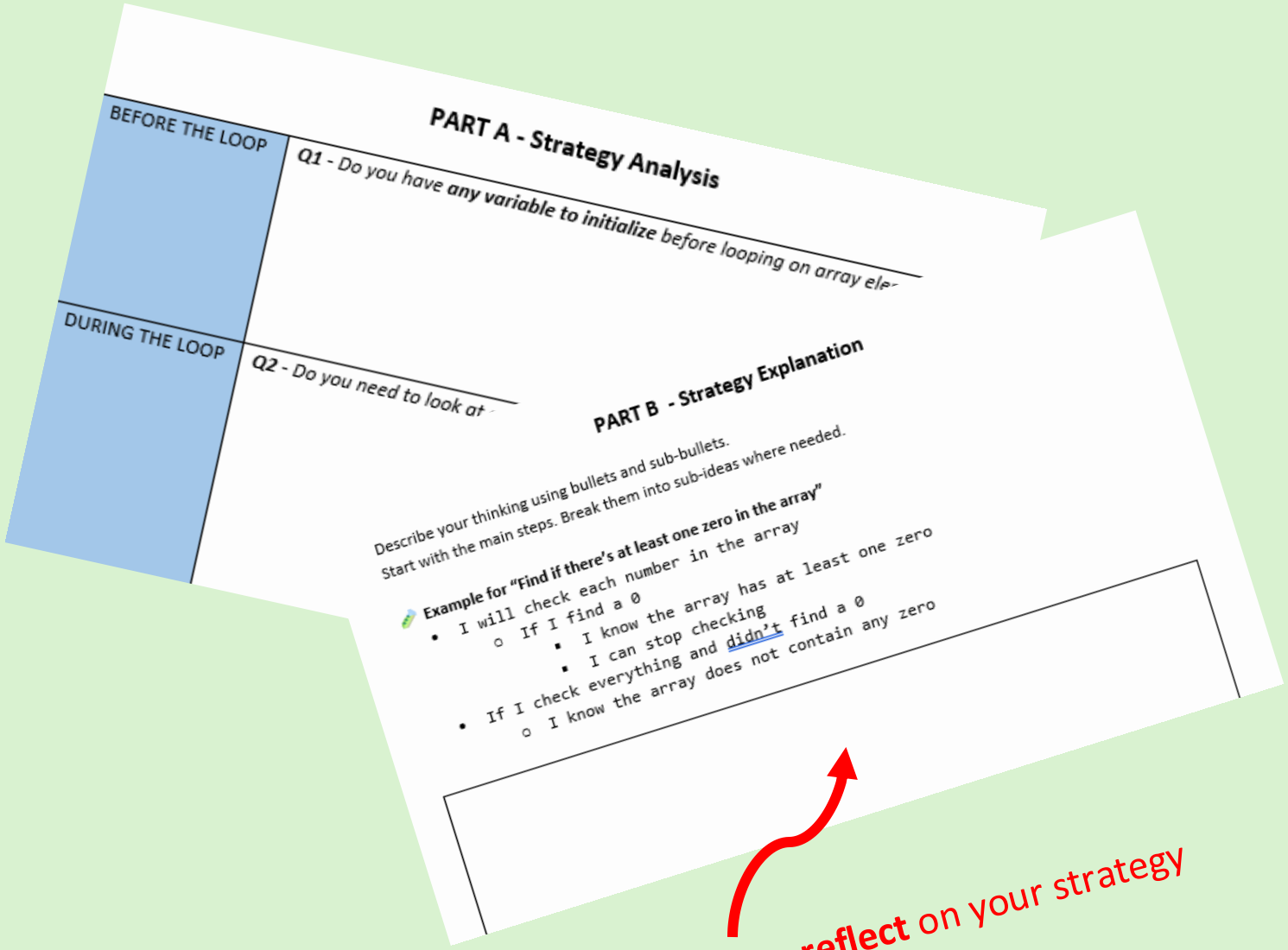
A Yes, it will work.

B No, it will not work.

Check something **on at least one element**

CHALLENGE 3

10 MIN



You don't **need to code**, but to **reflect** on your strategy

CHALLENGE 3 STRATEGY ANALYSIS

"We want to check that an array contains **at least** one number 3."

BEFORE THE LOOP	***Q1*** *- Do you have* ***any variable to initialize*** *before looping on array elements?* We initialize our variable (has_3) to false.
DURING THE LOOP	***Q2*** *- Do you need to look at* ***every element****, or* ***just some****?* We need to loop on every elements. ***Q3*** *- What are you* ***comparing or counting****?* We are checking if each element is equal to 3 or not. ***Q4*** *- What* ***condition would break*** *the rule or stop your check early?* If 1 element is equal to 3, our variable (has_3) is set to true. We can break our loop.
AFTER THE LOOP	***Q5*** *- What do you need to do after the loop?* We print the result depending on our variable (has_3).

CHALLENGE 3 STRATEGY PLAN

Describe your thinking using **bullets and sub-bullets**.
Start with the main steps. Break them into sub-ideas where needed.

- We define a boolean variable (has_3)
- We set has_3 to false

- We go through the array elements
  - For each number, we check if it's a 3
  - If yes, we set has_3 to true
    - We can stop the loop, since we know the answer

- We print the result depending on has_3

CHALLENGE 3 QUIZ

Algorithm Optimization

Complete the missing condition

CODE A

```
INPUT array
SET hasA3 TO False

FOR index FROM 0 TO 4
        IF array[index] == 3
                hasA3 = True



PRINT hasA3?  “at least a 3" : “no 3"
```

CODE B

```
INPUT array
SET hasA3 TO False

SET index TO 0
WHILE ____________________
        IF array[index] == 3
                hasA3 = True
        index++

PRINT only3?  "only 3" : "not only 3"
```

CHALLENGE 3 QUIZ

Algorithm Optimization

ANSWER

CODE A

```
INPUT array
SET hasA3 TO False

FOR index FROM 0 TO 4
        IF array[index] == 3
                hasA3 = True



PRINT hasA3?  “at least a 3" : “no 3"
```

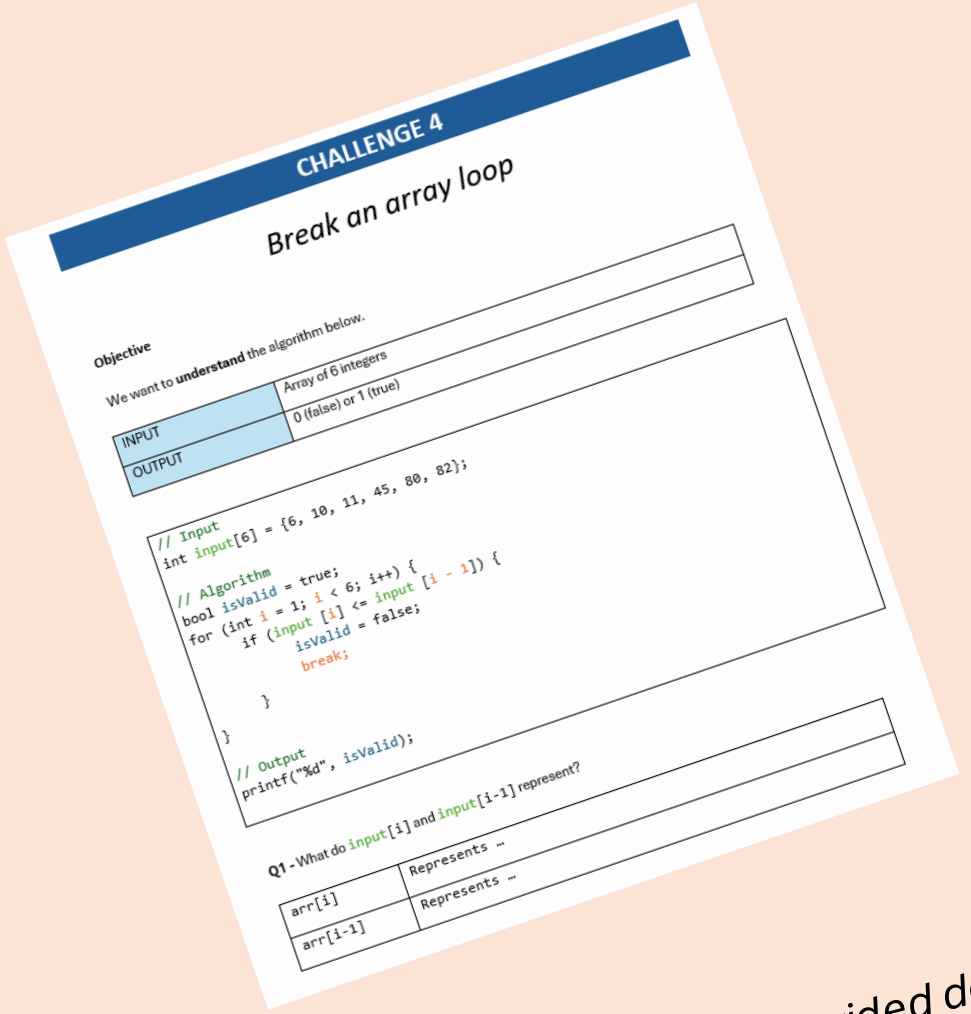
CODE B

```
INPUT array
SET hasA3 TO False

SET index TO 0
WHILE (index < 4 AND NOT hasA3)
        IF array[index] == 3
                hasA3 = True
        index++

PRINT only3?  "only 3" : "not only 3"
```

Break
an array
loop

CHALLENGE 4

10 MIN



CHALLENGE 4

Break an array loop

Objective

We want to **understand** the algorithm below.

INPUT	Array of 6 integers
OUTPUT	0 (false) or 1 (true)

```
// Input
int input[6] = {6, 10, 11, 45, 80, 82};

// Algorithm
bool isValid = true;
for (int i = 1; i < 6; i++) {
    if (input [i] <= input [i - 1]) {
        isValid = false;
        break;
    }
}

// Output
printf("%d", isValid);
```

Q1 - What do `input[i]` and `input[i-1]` represent?

arr[i]	Represents …
arr[i-1]	Represents …

Work on the provided doc

CHALLENGE 4 CORRECTION

Q1 - What do input[i] and input[i-1] represent?

arr[i]	Represent the current iteration element
arr[i-1]	Represent the previous iteration element

CHALLENGE 4 CORRECTION

Q2 - Execute mentally this code and write the output for each input

```
// Input
int input[6] = {6, 10, 11, 45, 80, 82};

// Algorithm
bool isValid = true;
for (int i = 1; i < 6; i++) {
        if (input [i] <= input [i - 1]) {
                isValid = false;
                break;
        }
}

// Output
printf("%d", isValid);
```

{1, 2, 4, 7, 8, 9}	1
{1, 2, 7, 4, 8, 9};	0
{1, 2, 3, 4, 5, 0};	0

CHALLENGE 4 CORRECTION

Q3 - Understand **the goal on this algorithm**, by completing the comment below

```
// Input
int input[6] = {6, 10, 11, 45, 80, 82};

// Algorithm
bool isValid = true;
for (int i = 1; i < 6; i++) {
        if (input [i] <= input [i - 1]) {
                isValid = false;
                break;
        }
}

// Output
printf("%d", isValid);
```

The goal of this algorithm is to check

Whether or not the list of numbers is **ordered from smallest to largest number.**

CHALLENGE 4 CORRECTION

Q4 - Update the previous exercise code to avoid the usage of a BREAK inside the LOOP

BEFORE

```
// Input
int input[6] = {6, 10, 11, 45, 80, 82};

// Algorithm
bool isValid = true;
for (int i = 1; i < 6; i++) {
        if (input [i] <= input [i - 1]) {
                isValid = false;

                break;
        }
}

// Output
printf("%d", isValid);
```

AFTER

```
// Input
int input[6] = {6, 10, 11, 45, 80, 82};

// Algorithm
bool isValid = true;
int i=1;
while (i < 6 && isValid) {
        if (input [i] <= input [i - 1]) {
            isValid = false;
         }
        i++;
}

// Output
printf("%d", isValid);
```

Congrats !

- ✓ Be able to compare **previous and next array** elements
- ✓ Check a condition on **every array element**
- ✓ Check a condition on **at least 1 array** element
- ✓ Understand the different approaches **to break a loop**

RESSOURCES FOR THIS COURSE

BOOKS

Our course book: C Programming for the Absolute Beginner
https://shorturl.at/8b42f

ONLINE LEARNING

https://www.w3schools.com/c
https://www.programiz.com/c-programming/getting-started
https://www.learn-c.org/en/Welcome

TOOLS

Code in C on your computer
https://code.visualstudio.com/download

Code in C online
https://www.programiz.com/c-programming/online-compiler

Debug in C online
https://pythontutor.com/c.html#mode=edit

Design flowchart and execute it
http://www.flowgorithm.org/download/index.html

https://www.coursera.org/learn/programming-c
https://www.w3schools.com/c/
https://codecombat.com/play
https://www.codingame.com/ide/puzzle/onboarding