CS1: Challenge 3 (Arrays & Invariants)

Simon Liénardy Benoit Donnet

1 Problem Description

Let `A` and `B`, two arrays of integers of size resp. `N` and `M`.The values in the arrays are in strict ascending sorted order[1]. You are asked to place, in a third array `C`, the common elements to `A` and `B`. The values in the array `C` must be placed, too, in strict ascending sorted order. You must consider that the array `C` will always be large enough.

Example: If `N` equals 5 and `M` equals 7, the arrays `A` and `B` may look like this:

	0				4
A:	3	5	6	9	10

	0						6
B:	2	3	4	6	7	8	9

The resulting array `C` would be the following (`?` denotes an undetermined value):

	0											11
C:	3	6	9	?	?	?	?	?	?	?	?	?

Be careful! While solving this problem, you must follow those constraints:

- you can only use a single while-loop;
- the theoretical complexity must be $\mathcal{O}(M+N)$. Moreover, your program cannot have more than $M+N$ iterations. The iterations number will be checked!
- you are not asked to print something on stdout. Doing so may make the correction step fail.

The code template is the following:

```c
#include <stdio.h>

int main(){
    const unsigned int N = ...;
    const unsigned int M = ...;
    const unsigned int L = ...; /* guaranteed large enough */
    int A[N];
    int B[M];
    int C[L];

    /*
     * Filling A and B (non-disclosed code)
     */

    // Your code will be inserted here.

}//fin programme
```

Consider that the code you submit will be copied and pasted at the line where the comment "`Your code will be inserted here`" is written. Henceforth, declare again the constants `N`, `M`, or `L` or

[1] Hence, each element in an array is unique.

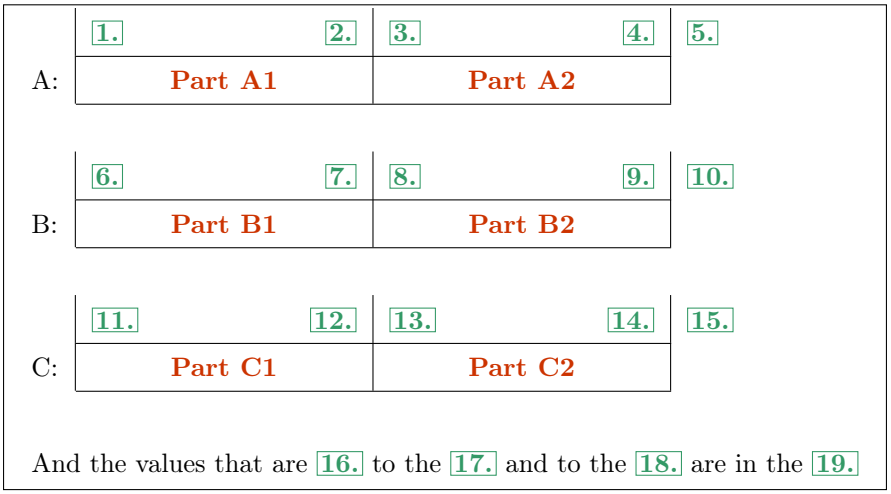
else the arrays `A`, `B`, or `C` will make the compilation fail (and, subsequently, a null mark for that Challenge submission).

In your submission, you must provide your code, as well as the Graphical Loop Invariant and the Loop Variant you used to derive it. The formatting of the Graphical Loop Invariant is detailed in Sec. 2, while the way to encode your Loop Variant is described in Sec. 3.

Use the template supplied with these instructions.

2 Submitting an Invariant

Here is a blank Graphical Loop Invariant:



This Graphical Loop Invariant must represent the 3 arrays manipulated by your program. The boxes 1. to 15. should be used to mention arrays indices. The boxes should be replaced by constant or variables names.

You do not have to replace each box: if you precise that the value for the box 9. should be "k", it is implicit that the value for box 10. should be "$k + 1$": then there is no need to write so.

For the others boxes, the available choices are sum up in the Table 1.

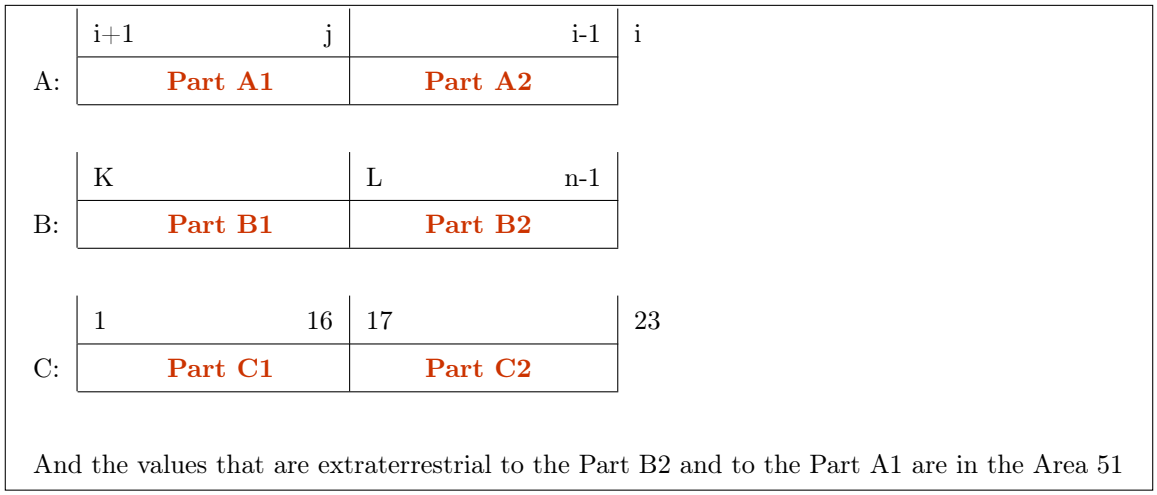
Box 16	**Boxes 17, 18 et 19**
1. different;	1. Part A1
2. maximal;	2. Part A2
3. browsed;	3. Part B1
4. analyzed;	4. Part B2
5. congruent;	5. Part C1
6. extraterrestrial;	6. Part C2
7. common.	7. Area 51

Table 1: Invariant: Available choices for the boxes **16** à **19**.

Please indicate, in the template, for the answer to the Graphical Loop Invariant part, the number of the box, followed by a dot, followed by the numerical value or the name of the constant or the variable of your choice (for the boxes 1. to 15.) or a number between 1 and 7 that corresponds to your choice (for the boxes 16. to 19.).

2.1 Example

If you think that a good Graphical Loop Invariant should be the following[2]



Then, fill in the answer template as follows:

```
1. i+1
2. j
3. _
4. i-1
5. i
6. K
7.
8. L
9. n-1
10.
11. 1
12. 16
13. 17
14.
15. 23
16. 6
17. 4
18. 1
19. 7
```

One can see that the variables and constants may be affected with a +1 or -1 operation. One just has to write this +1 or -1, as shown above. Do not insert brackets.

In all cases, if you think thta a good answer consists in leaving a box blank, do not write anything, or a "_" symbol.

For your convenience, the numbers of the 19 boxes are already written in the template to fill.

3 Loop Variant

The Loop Variant[3] enables to find a termination proof. You are asked to provide the Loop Variant of your loop.

[2]This is an example to precise how to encode an Invariant. I added some bullshit on purpose. Note that "Part" and "Area" come from the same French term "Zone": there is an obvious play of words in French.

[3]Not to be confused with the loop condition or the condition upon which the loop must be stopped.

In the template, at the question addressing the Loop Variant, we ask you to express it as a valid C expression.

The correction will check:

- That this expression uses the variables present in your code;
- That this is an Integer expression, positive if the loop condition is true;
- That the value of the expression is strictly decreasing from an iteration to another.

3.1 Example

If you think that the proper Loop Variant for your code (that uses the variable `foo` and the constant `BAR`) should be:

$$t = \texttt{BAR} + 17 - \texttt{foo}$$

Please encode:

```
BAR + 17 - foo
```

Do not add `"LV ="` or `"t ="`. Please only provide the expression that can be used to compute the value of your Loop variant.