Advanced C# – Debugging

The goal of this lab is to practice **debugging techniques** in scenarios where a piece of code does not work correctly. Your task is to pinpoint the bug and fix it (without rewriting the entire code).

Problem 3. Be Positive

You will receive some sequences of numbers on the console; your task is to **remove all negative numbers** and print back each sequence.

On the first line of input you are given a **count N – the number of sequences**.

On each of **the next N lines** you will receive some **numbers surrounded by whitespaces**.

You need to check each number, if it's positive – print it on the console; if it's negative, add to its value the value of the next number and only **print the result if it's not negative**. You only perform the addition once, e.g. if you have the sequence: -3, 1, 3, the algorithm is as follows:

- -3 is negative => add to it the next number (1) => -3 + 1 = -2 still negative => do not print anything (and don't keep adding numbers, you stop here).
- The next number we consider is 3 which is positive => print it.

If no numbers can be obtained in this manner for the given sequence, print **"(empty)".**

Example:

Input	Expected Output	Comments

`3`
`  3 -4     5 2   123`
`-1 -1 3 4`
`-2         1`
	`3 1 2 123`
	`3 4`
	`(empty)`
		`(3) (-4 + 5 = 1 > 0) (2) (123)`
		`(-1 + (-1) < 0) (3) (4)`
		`(-2 + 1 < 0)`

Output

Print on the console **each modified sequence on a separate line.**

Constraints

- The **number N** will be an integer in the range [1 … 15].
- The **numbers in the sequences** will be integers in the range [-1000 … 1000].
- The **count of numbers in each sequence** will be in the range [1 … 20].
- There may be **whitespaces anywhere around the numbers** in a given sequence

Tests

Input	Program Output	Expected Output

`3`
`  3 -4     5 2   123`
`-1 -1 3 4`
`-2         1`
	`Exception…`	**`3 1 2 123`**
		`3 4`
		`(empty)`

`1`
`0 -2 2 -2 3`
	`Exception…`	**`0 0 1`**