Prof. Dr. Annette Bieniusa
Philipp Lersch, M. Sc.

RPTU Kaiserslautern-Landau
Fachbereich Informatik
AG Softwaretechnik

Exercise 2: Programming Distributed Systems (Summer 2025)

Submission Deadline: 13.05.25 AOE

- You need a team and a Gitlab repository for this exercise sheet.
- In your Git repository, create a branch for this exercise sheet (for example with `git checkout -b ex2`)
- Create a folder named "ex2" in your repository and add your solutions to this folder.
- Test your submission with the provided test cases[1]. Feel free to add more tests, but do not change the existing test cases.
- Make use of Dialyzer annotations to improve your code quality

1 Time and causality

- Give an example execution that shows that for two timestamps from Lamport clocks, $C(e_1) < C(e_2)$ does not imply that $e_1 \rightarrow e_2$.
- Given a configuration of three servers S^1, S^2, S^3 and their current corresponding vector-clocks $S^1_C = [1, 3, 2]$ $S^2_C = [1, 4, 2]$ $S^3_C = [1, 0, 3]$. Is such a configuration possible? If yes, draw an example execution. If no, explain why.

2 Implementing Vector Clocks

A vector clock is a mapping from processes to positive integers[2]. Implement a module named `Vectorclock` with the following functions:

- `new()` creates a new vector clock, where all processes have value 0.
- `increment(vc, p)` increments the entry of process `p` by 1.
- `get(vc, p)` returns the value for process `p`.
- `leq(vc1, vc2)` checks, whether `vc1` is less than or equal to `vc2`. This is the case, iff $\forall p.\ get(vc_1, p) \leq get(vc_2, p)$.
- `merge(vc1, vc2)` merges two vector clocks by computing their least upper bound (the smallest vector clock `v`, such that $vc_1 \leq v$ and $vc_2 \leq v$).

Feel free to add more tests, but do not change the existing test cases.

[1] You can use `mix test` for executing the tests

[2] In the literature it is often assumed that processes are numbered which allows to write down clocks like $[4, 7, 3]$ or $\begin{pmatrix} 4 \\ 7 \\ 3 \end{pmatrix}$ instead of the longer $\{p_1 \mapsto 4, p_2 \mapsto 7, p_3 \mapsto 3\}$. However, in this exercise we do not assume that the number of processes is known and arbitrary terms can be used as process names.

3 Testing Vector Clocks

Specify at least 3 invariants that should hold for your vector clock implementation. The tests should be using these invariants, derive at least 2 test cases for each invariant and for the following example invariant:

1. $new() === new()$