### What are the necessary conditions for deadlocks (discussed in the lecture) [0.5 points]?

**Mutual Exclusion**: A limited number of threads are concurrently utilizing a resource

**Hold and wait:** A thread is holding a resource and waits for additional resources which are held by other threads

**No preemption:** Waiting for a resource that can only voluntarily be released by the thread holding the resource (no stealing involved)

**Circular wait:** Each thread is waiting for a resource that is being held by another and so on. E.g. Thread 1 waits for a resource held by Thread 2, Thread 2 waits for a resource held by Thread 3, Thread 3 waits for a resource held by Thread 1.

### Why does the initial solution lead to a deadlock (by looking at the deadlock conditions) [0.5 points]?

A circular wait may occur, because it is possible that every philosopher stops thinking at the same time, which leads to the situation the everyone takes the first fork before anyone else can take the second fork. As a result, every philosopher waits for the philosopher next to him to release their fork, which doesn’t happen.

### Does the strategy described in subtask 2 resolve the deadlock and why [1 point]?

### Measure the time spent in waiting for fork and compare it to the total runtime [3 points].

Testantwort

### Can you think of other techniques for deadlock prevention?

Testantwort

### Make sure to always shutdown the program cooperatively and to always cleanup all allocated

### resources [2 points]

Testantwort