Team:

- Aljoscha Alquati - se21m019
- Michael Goll - se21003

Subtask 1: A naive implementation

With long waiting periods (around 1000ms) deadlocks do not occur (at least in a reasonably long observation period). A deadlock was encounterd with 3 philosophers and waiting periods (Thinkingtime and Eatingtime) of 10 ms.

The shorter the waiting periods, the more likely was a deadlock (all philosophers are waiting for forks).

Subtask 2: Deadlock prevention

Why does the deadlock occur? Answer the following questions:

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

    1. Mutual exclusion
    2. Hold and wait
    3. No preemption
    4. Curcular wait

- **Why does the initial solution lead to a deadlock (by looking at the deadlock conditions) [0.5 points]? Hint: if you cannot provoke a deadlock add sleep's in order to make it more frequent (in the lecture we also had aribrary sleeps)**

    Deadlocks occur, because all four conditions are present in the naive solution.

    1. the forks are the limited resources which are requested concurrently
    2. the forks are held and other resources/forks are requested in the meantime
    3. no stealing resource locks by other threads
    4. like the philosophers sitting on a round table each waiting for a neighbors fork

- **Switch the order in which philosophers take the fork by using the following scheme: Odd philoso- phers start with the left fork, while even philosophers start with the right hand [6 points]. Make sure to use concurrency primitives correctly!**

    Implementiert

- **Does this strategy resolve the deadlock and why [1 point]?**

    Implementing the "Odd-Philosopher-Strategy" resolved the deadlock-issue, because for a deadlock to occur, all 4 conditions need to be present. By implementing this strategy, the "Circular waits" were removed.

- **Measure the total time spent in waiting for forks and compare it to the total runtime. Interpret the measurement - Was the result expected? [3 points].**

    5 Philosophers, 5 ms Thinkingtime, 5 ms Eatingtime: Total waittimes: 75.278,00ms / Total run time: 269.811,00ms

    10 Philosophers, 100 ms Thinkingtime, 100 ms Eatingtime: Total waittimes: 141.076,00ms / Total run time: 475.223,00ms

    Roughly a quarter is waiting time. That is rather surprisingly high.

- **Can you think of other techniques for deadlock prevention?**

    In this case, one could think of resolving this issue by allowing stealing of threats in some way. If philosophers would not have to wait until another philosopher has finished eating before starting to eat themselves and therefore removing the "No preemption" condition.

- **Make sure to always shutdown the program cooperatively and to always cleanup all allocated resources [2 points]**

    Implementiert