Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn
Philipp Bamberger

Distributed Systems, Summer Term 2019
Exercise Sheet 5

Exercise 1: Weak Agreement

You are given the following problem for two processes to find a weak agreement.

- Input/Output: Both processes have inputs from $\{0, 1\}$ and compute outputs from $\{0, 1, *\}$.
- Agreement: The outputs are equal or one output is $*$ (i.e., it is not allowed that one process outputs 0 and the other 1).
- Validity: If both inputs are the same, they need to output this value in case of no error.
- Termination: Non-failing processes need to terminate after a finite number of steps.

Design a wait-free algorithm/protocol that solves the given problem using only atomic read/write registers and prove the correctness of your algorithm (*hint:* there is a solution which uses only two registers).

Exercise 2: Consensus I

Alice and Bob live in the same town. Once a year they want to meet but they do not want to be seen together in public. So they want to meet at a secret place which one of them chooses. They know a wall in town which is painted white. In addition they know a painter who paints the wall in the color they wish and sends the person who gave him the order a *'before and after'* picture of the wall. (Of course they color-coded each possible meeting place with a single color in advance.)

1. Design an algorithm which ensures that Alice and Bob meet at the same place.
2. Can you expand your algorithm in such a way that it still works if Charlie wants to meet them as well?
3. How many persons can meet each other, if the wall is in front of the painters' shop and why? (You can assume that the painter immediately starts painting after receiving an order)

Exercise 3: Consensus II

A friend of yours is convinced to have found a great algorithm to solve consensus for 13 processes. His algorithm relies on a method called *'Fetch and Multiply'* which is described below. What would you tell him, if he asked you for your opinion?

```
public class RMW {
private int value;
public synchronized int FAM(int x) {
int prior=this.value;
this.value=this.value*x;
return prior;
}
}
```