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, \ast\}.
- **Agreement**: The outputs are equal or one output is \ast\ (i.e., it is not allowed that one process outputs 0 and the other 1), in case of no error.
- **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.

Sample Solution

We have two processes 1 and 2, where the input value of process 1 is \(x\), and the input value of process 2 is \(y\). Consider two atomic read/write registers A and B, which are initialized to value -1. Code for process 1:

```plaintext
write(A, x)
if (read(B) == 1-x) then
decide \ast
else
decide x
```

code for process 2:

```plaintext
write(B, y)
if (read(A) == 1-y) then
decide \ast
else
decide y
```

Now we show that all three conditions are satisfied.

- **Termination**: satisfied as each nonfailing process only does two steps and then takes action.
Observe that a process either decides to use its own input or \(*\). Note that \( \text{read}(A) == 1 - y \) only evaluates to true if process 1 has successfully written \( x \) and \( x \neq y \).

Next, to guarantee validity and agreement, we need to show that the following holds in the error free case.

- **Validity:** If both have the same input, then they have the same output.
  
  By the observation, it’s enough to show that both process don’t output a \(*\). This is true indeed, as one can only decide \(*\) if you know that the other process has a different input from yourself.

- **Agreement:** It enough to show that the combination \( \{0, 1\} \) is not possible. Assume it is possible. By validity, both process must have different inputs. It’s enough to show that at least one process outputs a \(*\) to reach a contradiction. Indeed, let us take a closer look at the two write operations of process 1 and 2. One of them is first, w.l.o.g. let the one of process 1 be first. Then process 1 will try and read what is written on the second register. There are two cases here, either it reads the input of the other process or -1. In the former, process 1 outputs a \(*\) and in the latter process 2 outputs the \(*\), since process 2 can now see the input of process 1 when it tries to read it afterwards.

**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).

**Sample Solution**

1. The algorithm looks as follows.

   Choose the color of the place you want to meet.
   Go to the Painter and instruct him to paint the wall in the corresponding color
   Look at the 'before and after picture' which you get from the painter.
   if the wall was white before
      the meeting place is the one you have chosen
   else
      the meeting place is the place according to the 'before color'

2. No, what Alice and Bob can do is known as the RMW primitive-swap, which has consensus number two (cf lecture).

3. If the wall is in front of the painters' shop it is basically the same as the RMW-primitive Compare and Swap (because you would see if the painter is already painting, or if someone is in the shop and instructing the painter to paint) which has consensus number \( \infty \). This means, that \( \infty \) many persons could meet.
   The Algorithm would look like this:
Choose the color of the place you want to meet.
Go to the Painter
Look at the wall in front of the painters’ shop
if the wall is white
enter the shop and instruct the painter to paint the wall in your color
the meeting place is the one you have chosen
else
the meeting place is the place according to the color of the wall

Exercise 3: Consensus II \hfill (2 Points)

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'. What would you tell him, if he asked you for your opinion? The method is described below.

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

Sample Solution

The presented procedure is commutative and thus the consensus number has to be smaller than or equal to two. So I can easily convince my friend that his algorithm does not solve consensus for 13 processes (by using a result from the lecture).