CvP - Werkcollege 12

Exercise 1 Consider two processes A and B that use a shared variable x, as below:

```
x := 5;

task A;
    x := 3 * x;
end A;

task B;
    x := x - 4;
end B;
```

Assume that the statement in the body of each process executes as a sequence of three atomic operations:

1. fetching the current value of x,
2. performing the specified arithmetic operation, and
3. storing the new value back in x.

Obviously, this code contains no explicit synchronization constructs, and the operations in the body of each process get executed concurrently with those of other processes.

(a) What values are allvpossible for x after the execution and termination of both processes?

(b) Which of the above values for x are acceptable as the intuitively correct outcomes of a complete run of the above application?

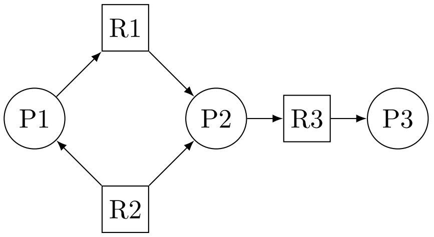
(c) What concurrency primitives, if any, need to be added to the above application to exclude incorrect outcomes?

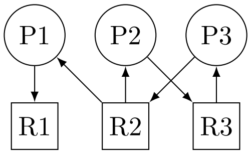
Exercise 2 Consider a concurrent system that consists of a (fixed) number of processes and resources (such as locks and semaphores). A *request/allocation graph* (RAG) is a directed graph whose vertices consists processes and resources. An edge $P \to R$ from a process P to a resource R denotes a *request* by process P for resource R. An edge $R \to P$ from a resource R to a process P denotes an *allocation* of resource R to process P. A RAG is an abstract view on the state of the system.

The execution of the system (i.e., state changes) is reflected on RAGs via graph transformations:

1. If process P requests resource R, we add an edge from P to R.
2. If a request $P \to R$ is granted, the direction of the arrow is flipped.
3. If process P deallocates resource R, we remove the edge from P to R.

Consider the following request/allocation graphs.





Suppose that no more requests are made. For each graph, determine whether or not they lead to a deadlock.