

# - Shared Memory -

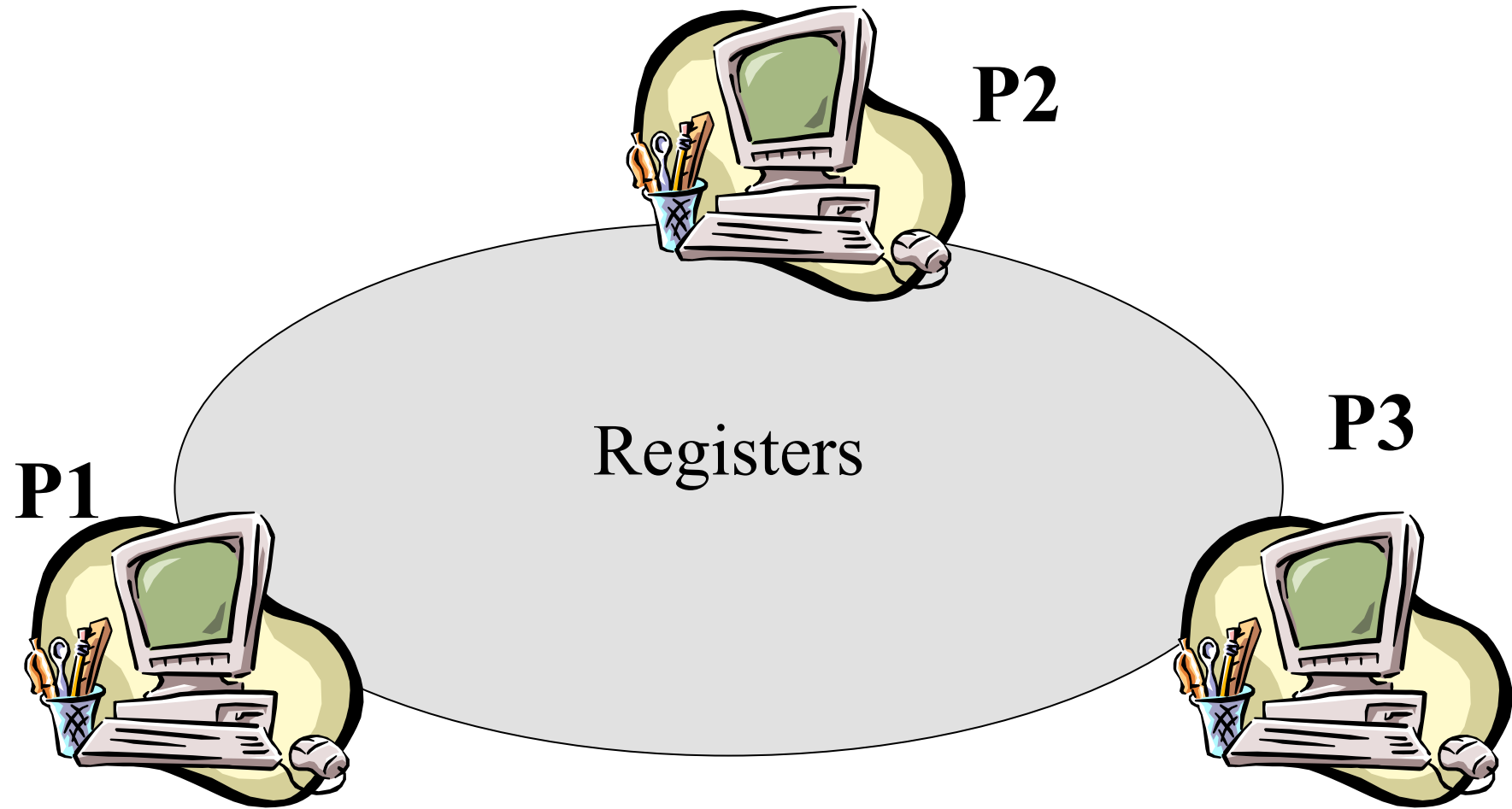
***R. Guerraoui***  
***Distributed Programming Laboratory***  
***lpdwww.epfl.ch***



© R. Guerraoui



# The application model



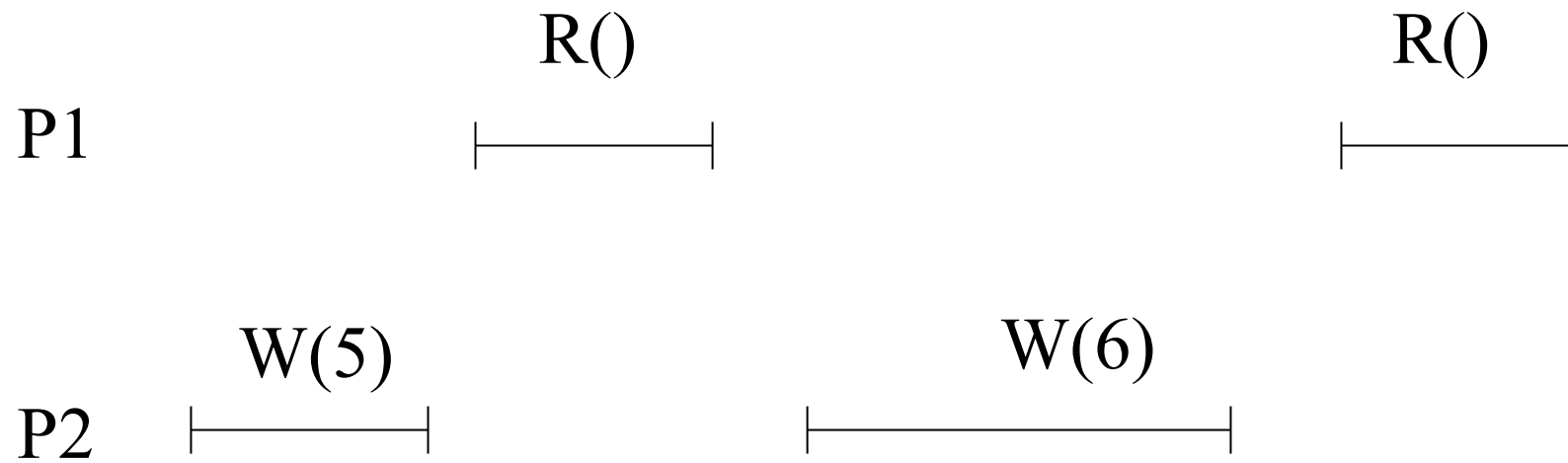
# Register (assumptions)

- For presentation simplicity, we assume registers of *integers*
- We also assume that the initial value of a register is 0 and this value is initialized (written()) by some process before the register is used
- We assume that every value written is *uniquely* identified (this can be ensured by associating a process id and a timestamp with the value)

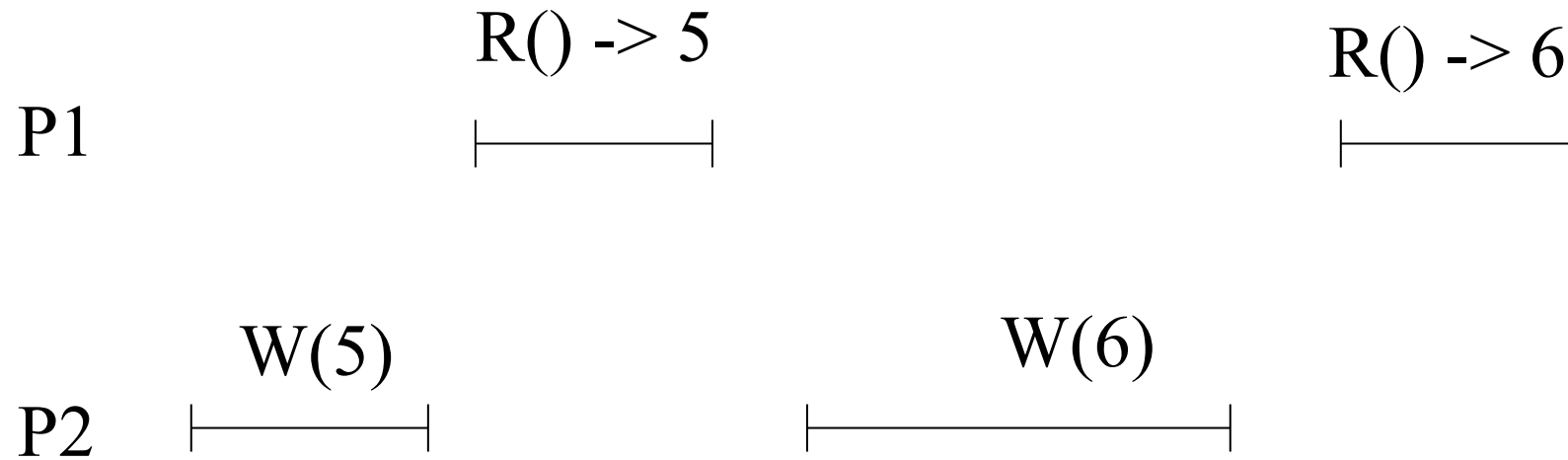
# Register: specification

- Assume a register that is local to a process, i.e., accessed only by one process:
- In this case, the value returned by a ***Read()*** is the last value written

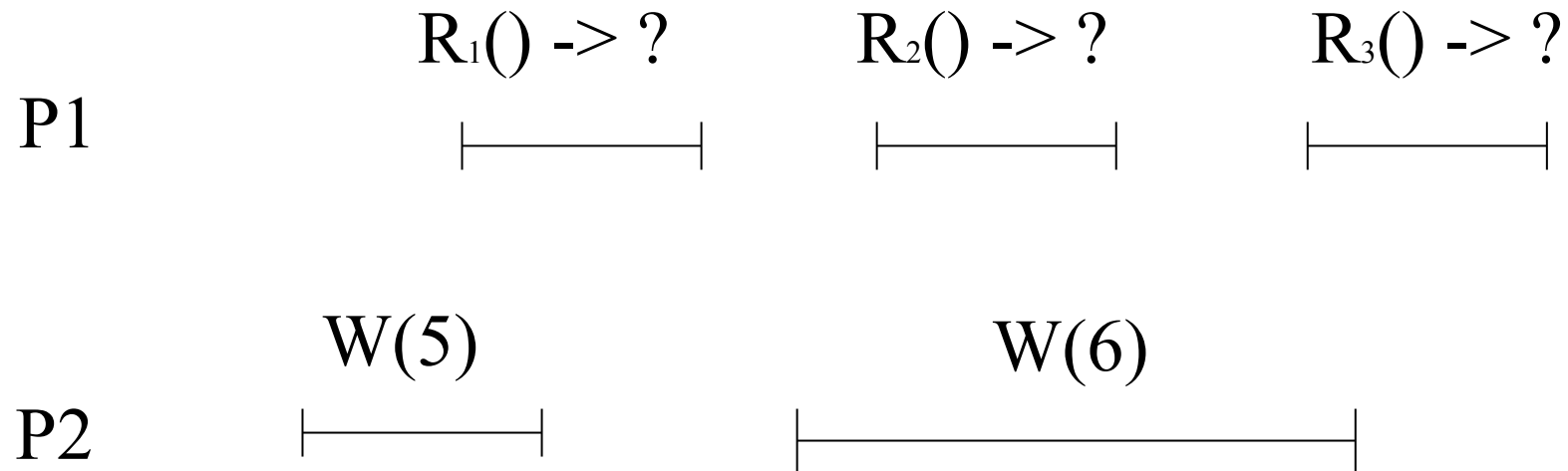
# Sequential execution



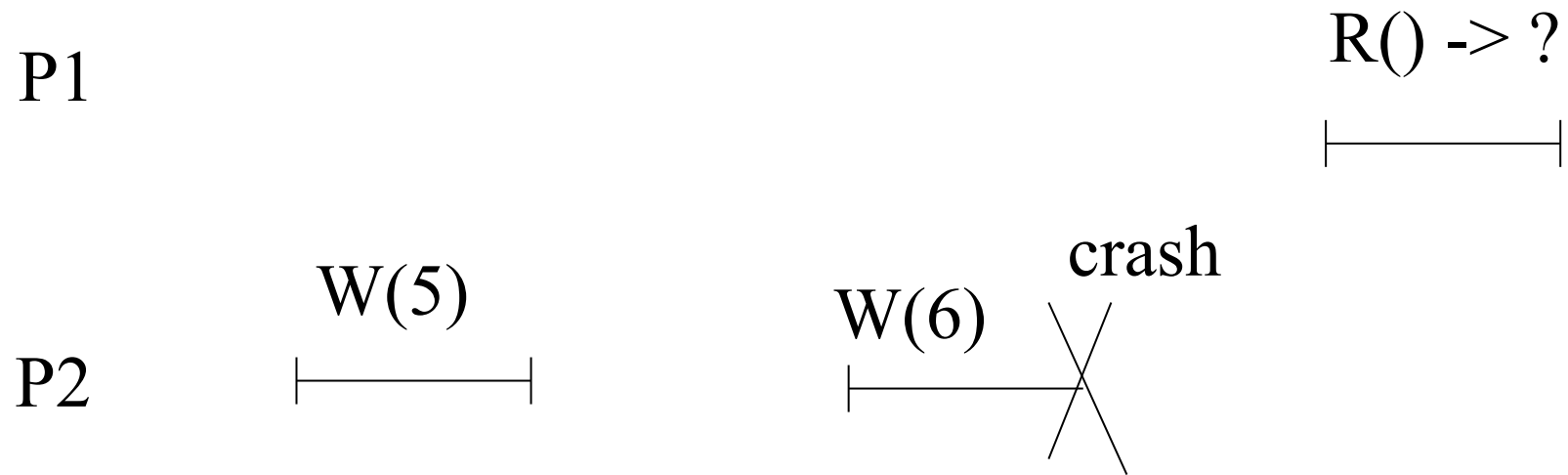
# Sequential execution



# Concurrent execution



# Execution with failures





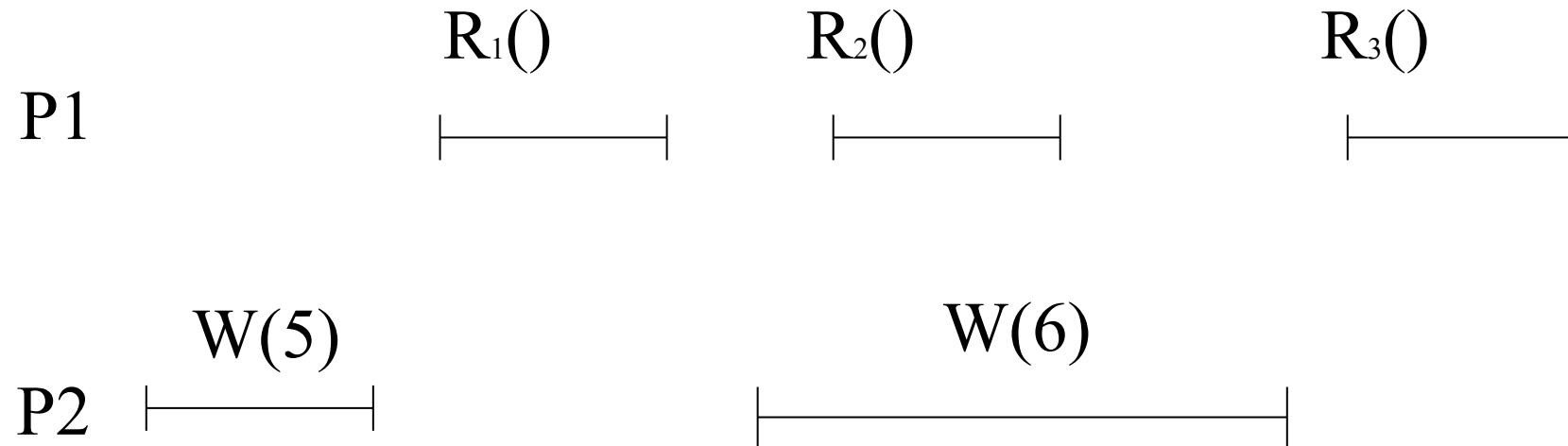
# Regular register

- It assumes only **one** writer; multiple processes might however read from the register
- It provides **strong** guarantees when there is no concurrent or failed operations (invoked by processes that fail in the middle)
- When some operations are concurrent, or some operation fails, the register provides **minimal** guarantees

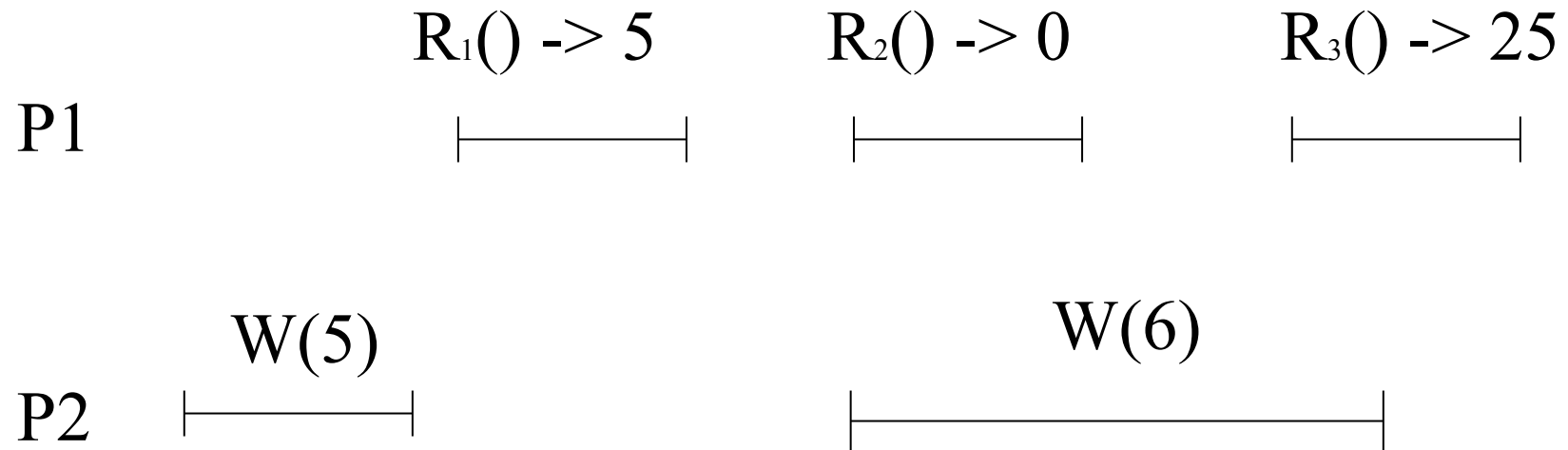
# Regular register

- ***Read()*** returns:
  - ✓ ***the last value*** written if there is no concurrent or failed operations
  - ✓ and otherwise the last value written or ***any*** value concurrently written, i.e., the input parameter of some ***Write()***

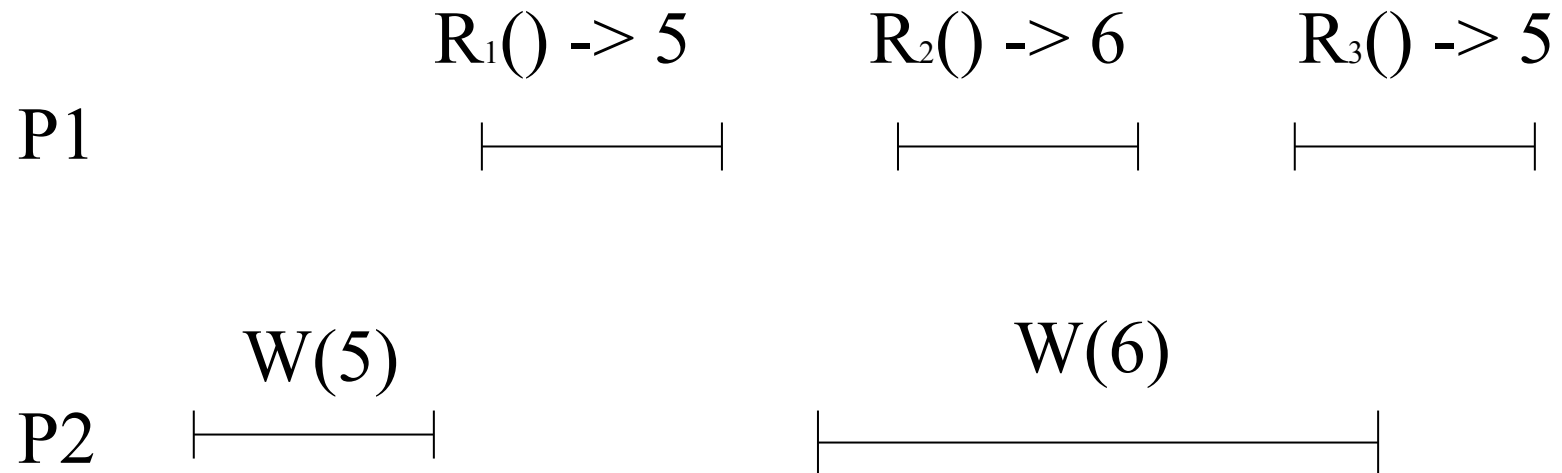
# Execution



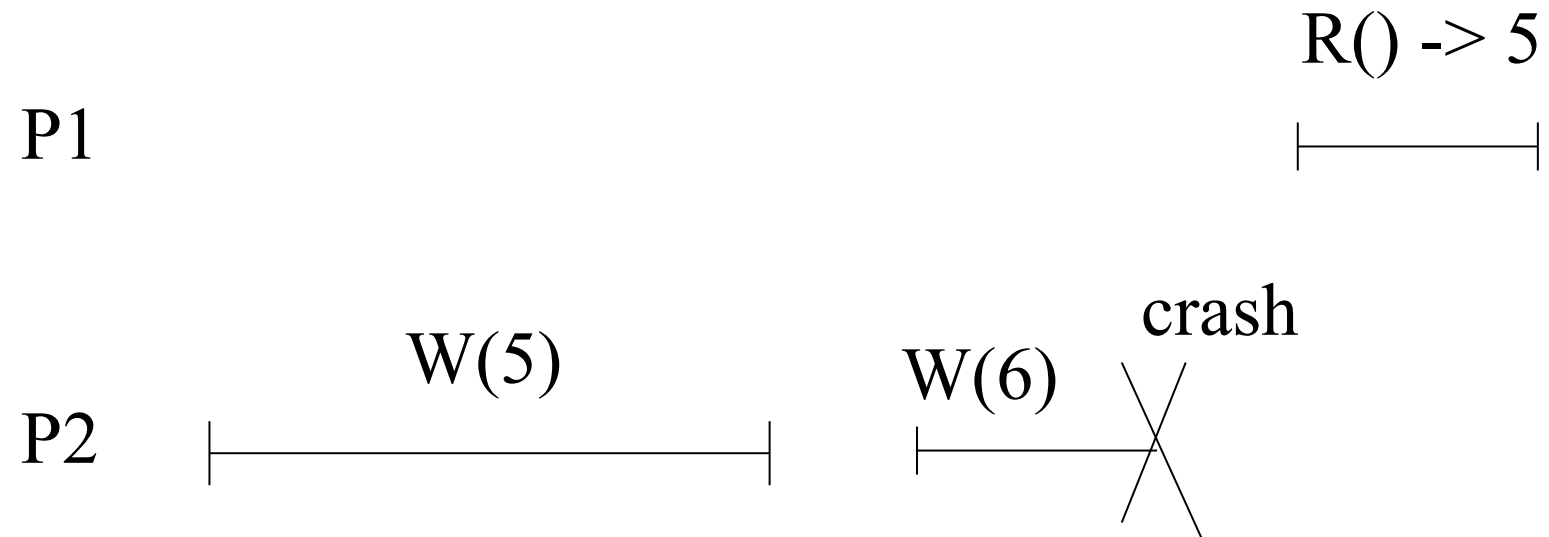
# Results 1



# Results 2



# Results 3



# Results 4



# Correctness

- Results 1: non-regular register (safe)
- Results 2; 3; 4: regular register