HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Benjamín Aage B. Birgisson

Exercise 2.1.1:

(a) No, this is not a valid transition, as it goes from „ready“ state to „suspended“ state and therefore should return back to the „ready“ state.

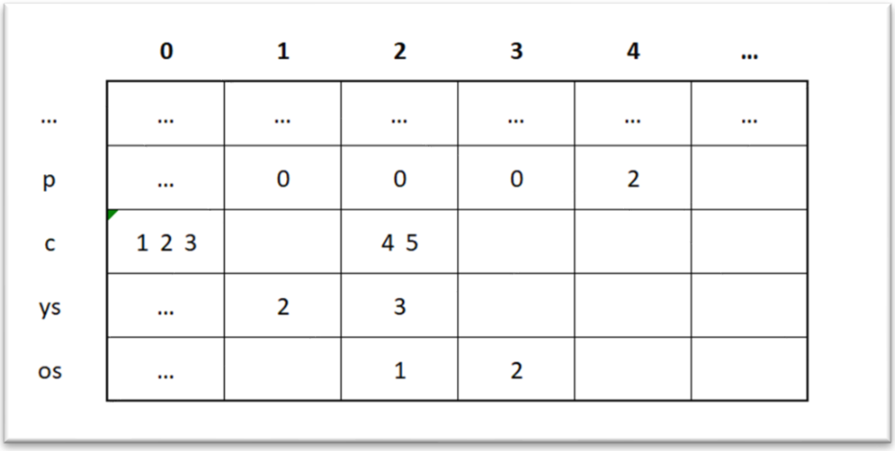
(b) Yes, this would be a valid transition between states (as long as the first move to „suspended“ state was from „blocked“ state, which is not shown here).

(c) No, this is not a valid transition as you can not go from “ready” state straight to “blocked” state.

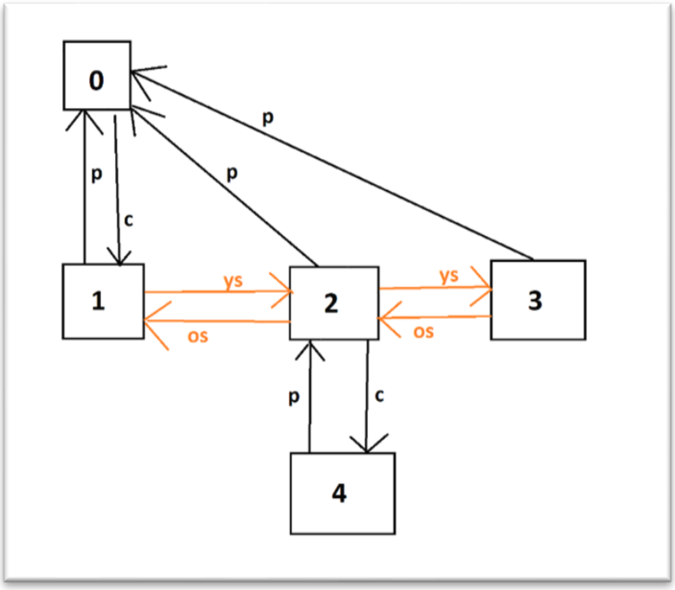
(d) No, this is not a transition, as you cannot move from a „ready“ state back to „new“ state (the process is only activated singularly, when it is ready to compete for the CPU, thereby not able to move back from “ready” to “new” state again).

Exercise 2.3.1:

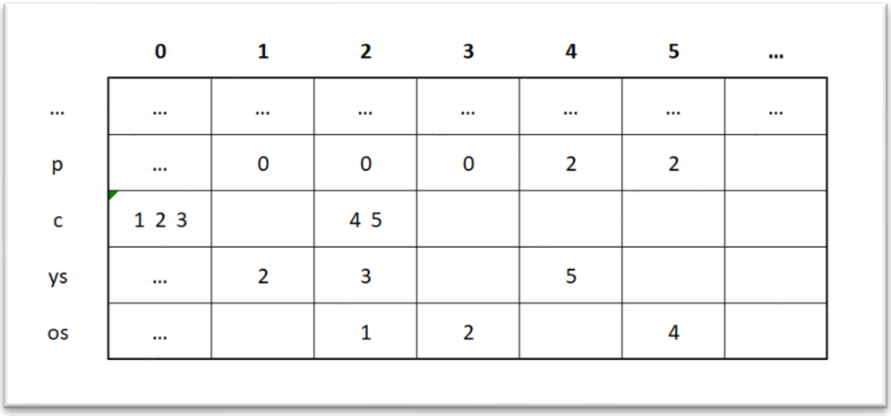
(a)



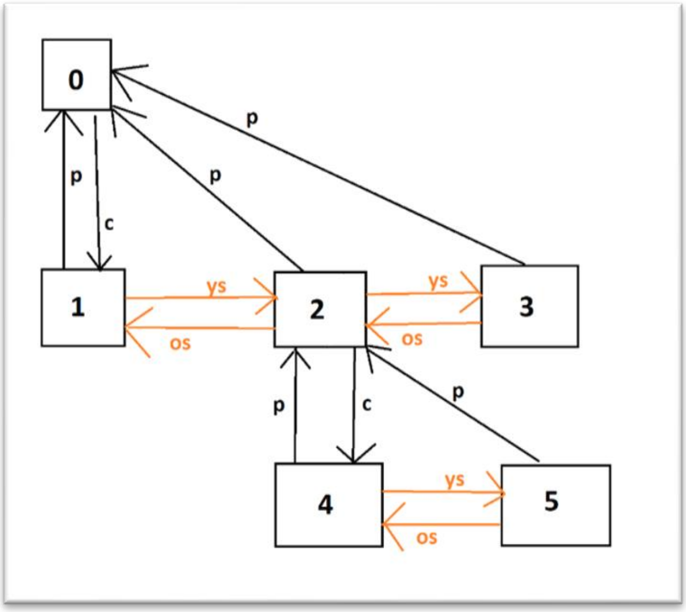
	0	1	2	3	4	...
...	...	...	...	...	...	...
p	...	0	0	0	2	
c	1 2 3		4 5			
ys	...	2	3			
os	...		1	2		



(b)

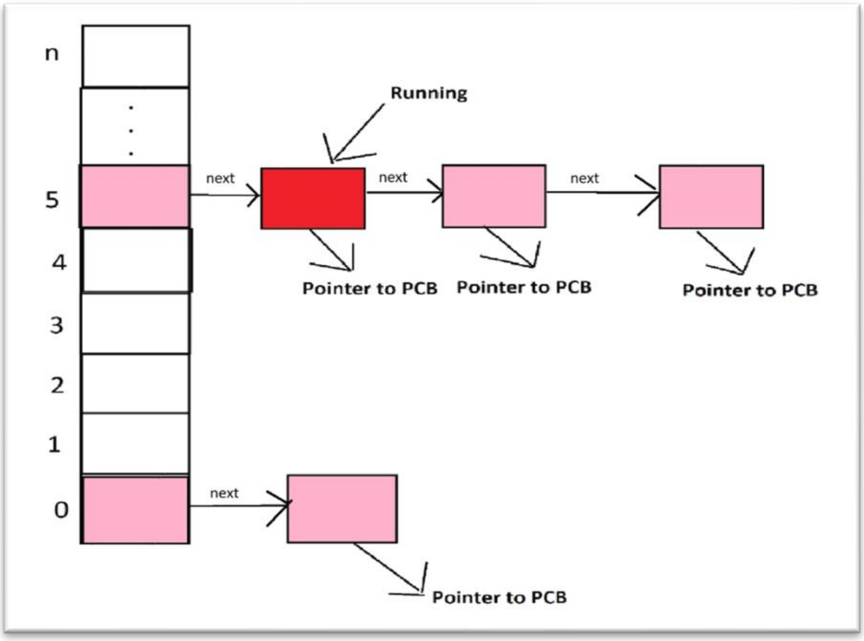


	0	1	2	3	4	5	...
...	...	...	...	...	...	...	...
p	...	0	0	0	2	2	
c	1 2 3		4 5				
ys	...	2	3		5		
os	...		1	2		4	



Exercise 2.3.2:

(a)



Exercise 2.4.1:

(a) You would have to add to “suspended(p)” the else-statement when the process state is “running” → “suspended_running” to clarify when a process was suspended during running state..

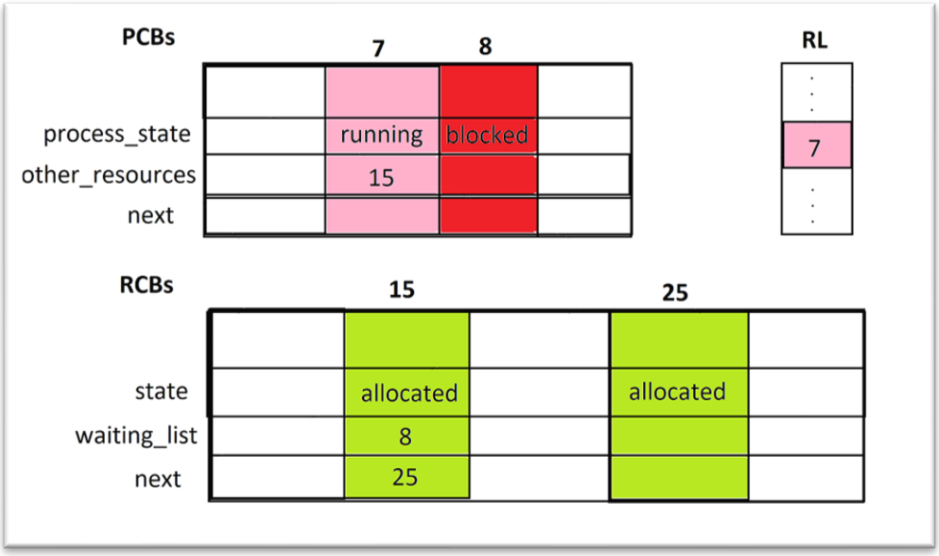
(b) Because the “activate(p)” function is checking if the process suspended was in a ready-state before suspension, and if so can then select the process to run next. However, the “suspend(p)” is only declaring the state in which a process was suspended, either the process was in “ready” state or in “blocked” state and therefore the “scheduler()” is not needed within that function.

(c) Because it is the job of the CPU to do so, as it allocates when a possible suspension occurs and/or when the process can again be activated again.

(d) It would else lead to an infinite loop, as the scheduler does never call the process to run.

Exercise 2.5.1:

(a)



PCBs

		7	8	
process_state		running	blocked	
other_resources		15		
next				

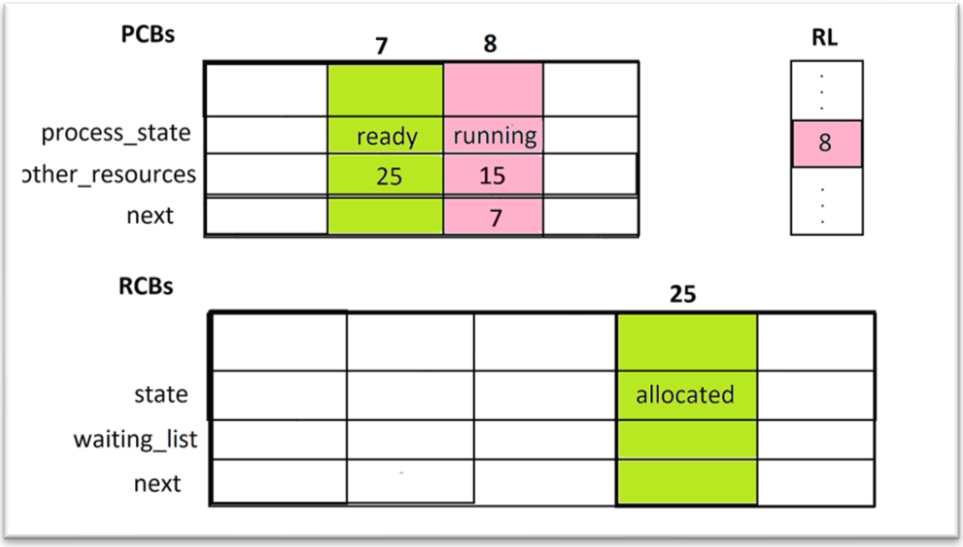
RL

⋮
7
⋮

RCBs

		15		25	
state		allocated		allocated	
waiting_list		8			
next		25			

(b)



PCBs

		7	8	
process_state		ready	running	
other_resources		25	15	
next			7	

RL

⋮
8
⋮

RCBs

				25	
state				allocated	
waiting_list					
next					

Exercise 2.6.1:

(a) I think it should be implemented as a single-threaded server, as 80% of all requests can be serviced without disk access, therefore only taking 15ms to process.

(b) 50%

Exercise 2.6.2:

(a) 3, 4, 6