CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
TDA384	Principles of concurrent programming			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
TDA384-0011-CDW		2022-08-18	5	4

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	14	
2	X	0	
3	X	16	
4	X	16	
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points		46	

4

a)

```
server(CurrentValue, [FirstWait|RestWait]) -> receive
    {signal} -> FirstWait ! semAcquired, //Assuming there are blocked threads when calling up().
                server(CurrenValue, RestWait);
    {wait, Pid} ->
        if
            CurrentValue == 0 -> server(CurrentValue, [FirstWait|RestWait]++Pid);
            currentValue /= 0 -> Pid ! semAcquired,
                                 server(CurrentValue - 1, [FirstWait|RestWait])
        end
    {value, Pid} ->
        Pid ! CurrentValue,
        Server(CurrentValue, [FirstWait, RestWait])
    end.

start(InitialValue) -> spawn(fun() -> server(InitialValue, []) end).
signal(Server) -> Server ! {signal}.
wait(Server) -> Server ! {wait, self()},
receive
    semAcquired -> proceed
end.
value(Server) -> Server ! {value, self()}
receive
    CurrentValue -> CurrentValue
end.
```

You used the signal immediately.

what about the server loop for the case that the list of waiting is empty?

6

4

6

b) Yes it is fair since blocked threads will be signaled in a FIFO manner. ✓

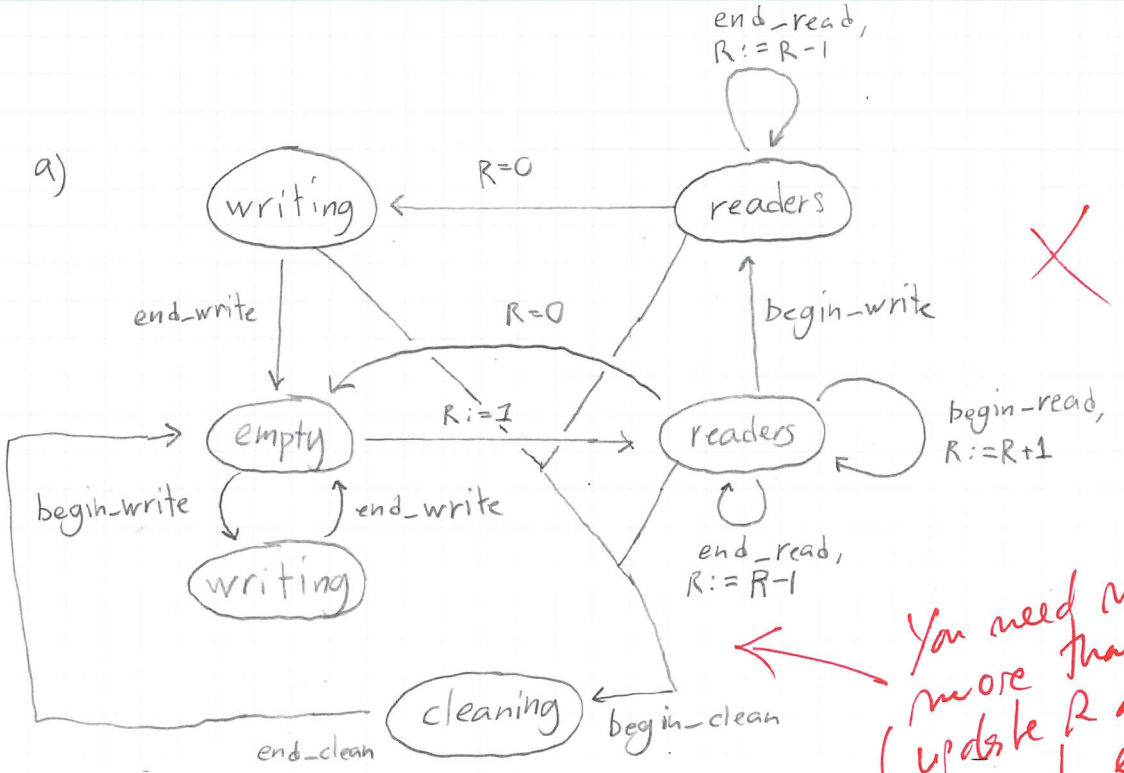
c) It won't pattern match with anything in the loop. Adding a "Ref" would make it work. That is writing this instead:

```
Ref = make_ref(),
Server ! {stop, self(), Ref, 0},
ok.
```

2

what about this waiting?

a)



X

You need much more than this! (update R and W for each end_read and end_write)

b) clean_board() → empty_board().

c) clean_board([], []) → empty_board();

clean_board([Writer|Rest], _) → Writer ! {cleaning, Ref},
clean_board(Rest, _);

X clean_board(_, [Reader|Rest]) → Reader ! {cleaning, Ref},
clean_board(_, Rest).

// Every reader/writer needs to handle the {cleaning, Ref}
// received by the clean_board.

begin_clean(Writers, Readers) → clean_board(Writers, Readers).

4

a) 1. False, in fine-grained locking both the pred and curr should be locked when using the find method. 2

2. True, by having more than one thread accessing the fine-grained it will have inconsistencies when validating. – False 0

b) This protocol isn't safe. +1 8

t_0 adds a node inbetween d and j, lets say e Example! +4

t_0 unlocks and terminates

t_1 locks, deletes node j, unlocks and terminates.

This would cause a problem because e would also be removed: 

~~3~~ +1

To fix this we would lock/unlock with the hand-over-hand method: Lock pred & curr → Unlock pred → Move pred & curr one step → Lock new curr. +2

6

c) Wrong usage ~~of the locks~~

Correct solution:

```
pred.lock(), curr.lock();          +2
while(curr.key < key) {
    pred.unlock();                 +2
    pred=curr;
    curr=curr.next();
    curr.lock();
}
```

Line 1, 2, 3, 12 & ~~14~~ 13 in the given code is the same.

4

a)

	state	new state if p moves	new state if q moves
s1	(2,2,f,f,f)	(3,2,t,f,f) = s3	(2,3,f,t,f) = s2
s2	(2,3,f,t,f)	(3,3,t,t,f) = s4	(2,5,f,t,t) = s5
s3	(3,2,t,f,f)	(5,2,t,f,t) = s6	(3,3,t,t,f) = s4
s4	(3,3,t,t,f)	(5,3,t,t,t) = s7	(3,5,t,t,t) = s8
s5	(2,5,f,t,t)	(3,5,t,t,t) = s8	(2,2,f,f,f) = s1
s6	(5,2,t,f,t)	(2,2,f,f,f) = s1	(5,3,t,t,t) = s7
s7	(5,3,t,t,t)	(2,3,f,t,f) = s2	(5,3,t,t,t) = s7
s8	(3,5,t,t,t)	(3,5,t,t,t) = s8	(3,2,t,f,f) = s3

4

b) Yes, not a single state that has both threads in critical section. 2

c) The flags aren't needed since compare and set already checks if one thread is in the critical section or not. 3

d) Remove all flags:

```
boolean turn = false;
```

p

```
while(true){
    while(!turn.CAS(false, true){};
    //CS
    turn = false;
}
```

q

```
// Same as p
```

4

e) Any number of threads. 3

16