

## Übung 5

---

### Aufgabe 5.1

a)

Thread 0	Thread 1	Variable (flag = false)
while (true) {		
while (flag != false) {		
	while(true) {	
	while (flag != false) {	
flag = true		flag = true
	flag = true	flag = true
critical_section()		
	critical_section()	

b)

Thread 0	Thread 1	Variable (counter = 0)
	while (true) {	
{	counter++	counter = 1
	if (counter == 3) {	
	}	
	}	
	while (true) {	
{	counter++	counter = 2
	if (counter == 3) {	
	}	
	}	
	while (true) {	
{	counter++	counter = 3
	if (counter == 3) {	
	critical_section	
while (true) {		
counter++		counter = 4
if (counter == 5) {		
}		
}		
while (true) {		
counter++		counter = 5
if (counter == 5) {		
critical_section()		

c)

Thread 0	Thread 1	Variable (mutex (unlocked))	Variable (mutex2 (unlocked))
Monitor.Enter(mutex)		mutex (locked by thread 0)	
	Monitor.Enter(mutex)		mutex2 (locked by thread 1)

Thread 0	Thread 1	Variable (semaphore [counter: 0])
while (true) {		
	while (true) {	
	if (semaphore.Wait(500)) {	
	semaphore.Release()	semaphore [counter: 1]
semaphore.Wait();		semaphore [counter: 0]
	}	
	}	
	while (true) {	
	semaphore.Release()	semaphore [counter: 1]
	while (true) {	
	if (semaphore.Wait(500)) {	semaphore [counter: 0]
	critical_section()	
critical_section()		

## Aufgabe 5.2

### a) Bounded Waiting

Turn wird mit 0 initialisiert, somit geht P0 nach der Prüfung der Schleife in den kritischen Bereich. P1 wird nun so scheduled das die Schleifenbedingung immer geprüft wird, wenn P0 im kritischen Bereich ist. Turn ist nun immer 0, falls P1 die Schleifenbedingung Prüft . P0 kann damit P1 beliebig oft überholen.

### b) Mutal Exclusion

flag[0] und flag[1] werden mit false initialisiert, somit ist es möglich das das P0 die Schleifenbedingung prüft und dabei unterbrochen wird, bevor flag[0] = true ausgeführt wird. P1 prüft dann auch die Schleifenbedingung und setzt flag[1] = true und geht in den kritischen Bereich. P1 wird durch P0 unterbrochen welches flag[0] = true und anschließend in den kritischen Zustand geht. Damit sind 2 Prozesse im kritischen Zustand.

### c) Progress

flag[0] und flag[1] werden mit false initialisiert. P0 setzt flag[0] = true, anschließend setzt P1 flag[1] = true. Nun befinden sich beide Prozesse in einer Endlosschleife da sie nun jeweils ihre Schleifenbedingung prüfen und noop ausführen.