# Musterlösung Dekkers Algorithmus

## Mutual Exclusion:

Wird im folgenden von „der while-Schleife“ gesprochen so ist immer die while (enter1) ... bzw. while(enter2)... Schleife gemeint.
Betrete oBdA P1 seine criticial section. D.h. dass im Schritt vorher die Bedingung der while-Schleife false war, d.h. enter2 war false. D.h. dass P2 zu dem Zeitpunkt wo P1 die Bedingung der while-Schleife überprüft hat entweder in seiner noncritical section war oder unmittelbar vor, hinter oder in der while (turn == 1) ... Schleife war. Bevor P2 in seine critical section kann, kommt er also auf jeden Fall noch zu dem Schritt vor er die Bedingung seiner while-Schleife überprüfen muss, d.h. er muss prüfen ob enter1 false ist. Da enter1 aber true ist und es auch so lange bleibt wie P1 noch in seiner critical section ist, kann P2 weder in der Zeit zwischen dem überprüfen von P1 seiner Bedingung seiner while-Schleife und dem Betreten von P1 von seiner criticial section noch wenn P1 in seiner criticial section ist, in die critical section.

## Eventual Entry:

Versuche oBdA P1 in die critical section zu kommen, d.h. P1 setzte enter1 auf true. Den nächsten Schritt den P1 dann irgendwann ausführt ist die überprüfung der Bedingung der while-Schleife. Ist enter2 false so betritt P1 seine critical section. Ang. enter2 ist also true. Dann ist P2 entweder in seiner critical section (a) oder versucht gerade selbst die critical section zu betreten (b). Betrachte zunächst den Fall (b), dass P2 auch gerade versucht seine criticial section zu betreten. Beide Prozesse versuchen also ihre critical section zu betreten, dabei verändern sie den Wert von turn nicht. turn ist also entweder 1 oder 2. Ist turn 1 so wird P2 irgenwann enter2 auf false setzen und dann in der while(turn == 1) .. Schleife so lange warten bis sich turn wieder geändert hat. Da P1, da turn 1 ist, nicht gerade in seiner while (turn == 2) .. Schleife ist, kommt P1 irgendwann dahin, dass er die Bedingung seiner while-Schleife überprüft und da enter2 false ist betritt P1 seine critical section. Ist turn 2 so ist es genau umgekehrt und P1 wartet in seiner while (turn == 2) .. Schleife und P2 kann seine critical section betreten. Irgendwann verlässt P2 seine critical section und setzt enter2 auf false und turn auf 1. Versucht P2 jetzt nicht in die critical section zu gehen während P1 seine nächsten beiden Schritte ausführt nämlich das Setzen von enter1 auf true und das überpüfen der Bedingung der while-Schleife so betritt P1 seine critical section. Versucht P2 aber direkt nach dem Verlassen seiner critical section diese wieder zu betreten so kann er endlich oft seine critical section betreten und wieder verlassen und zwar so lange bis P1 vom Scheduling her irgenwann mal wieder dran ist und enter1 auf true setzt. Dann wird P2 wenn er versucht das nächste Mal seine critical section zu betreten auf jeden Fall da turn 1 ist, enter2 auf false setzen und in seiner while (turn == 1) .. Schleife so lange warten bis turn wieder 2 ist. Zu betrachten bleibt noch der erste Fall (a), dass P2 gerade in seiner critical section ist. P2 wird die critical section irgendwann verlassen. P1 kann zu diesem Zeitpunkt auch schon weiter gelaufen sein. Ist turn 2 so kann es sein dass P1 irgendwo unmittelbar vor, nach oder in den ersten beiden Befehlen innerhalb des if-Blocks ist. Nachdem P2 seine critical section verlassen hat setzt er enter2 auf false und turn auf 1. Wie im Fall oben wird P1 dann irgendwann enter1 auf true setzen und in der Zeit bis dahin kann P2 theoretisch endlich oft seine criticial section betreten und verlassen. Nachdem P1 aber enter1 auf true gesetzt hat, kann er als nächstes die critical section betreten wie oben erklärt. Der zweite Fall ist, dass entweder turn nicht 2 ist oder P1 noch nicht so weit gekommen ist in der if-Abfrage zu überprüfen ob turn 2 ist. Dann ist enter1 noch true und P1 kann auf jeden Fall als nächstes die critical section betreten, denn auch wenn P2 direkt nach dem Verlassen seine critical section wieder versucht die critical section zu betreten wird er da enter1 true ist und turn 1 enter2 auf false setzen und in seine while(turn == 1) .. Schleife gehen.
Aus den Erklärungen folgt auch, dass ein Prozess endlich oft (bei fairem Scheldung aber nicht unendlich oft) überholt werden kann.

## Deadlock:

Ein Deadlock kann nur Auftreten wenn ein Prozess beim Versuch die critical section zu betreten unendlich lange in seiner while-Schleife bleibt. Wegen der gerade gezeigten Eventual entry wird er aber irgenwann seine critical seciton betreten, es kann also keinen Deadlock geben.

## Unnecessary Delay:

oBdA versuche P1 seine critical section zu betreten und P2 sei in seiner non criticial section oder habe terminiert. Dann ist enter2 false und turn 1. Ist P1 in seiner while (turn == 2) .. Schleife so wird er diese also irgendwann verlassen. In jedem Fall ist er also irgendwann an der Stelle wo er die Bedingung seine while-Schleife überprüft und da enter2 false ist betritt er seine critical section dann.