Name: _____

Übung zum Rundlaufverfahren mit Ankunftszeiten und Verklemmungen

① Man betrachte die vier zyklisch arbeitenden Prozesse A, B, C und D.
Die folgende Tabelle enthält die Prozesse mit ihren jeweils benötigten CPU-Zeiten und E/A-Zeiten, wobei ein CPU-Stoß und ein E/A-Stoß vollständig auszuführen ist:

	CPU	E/A
A	5	3
B	6	6
C	3	5
D	4	5

Als Zeiteinheit sei hier die Einheit ms zu nutzen. Zum Zeitschritt 0 steht der Prozess A auf der Bereit-Liste. Die Prozesse C und D werden erst ab Zeitschritt 5 zyklisch ausgeführt und B steht erst ab dem Zeitschritt 10 zur Verfügung.

Im folgenden mittels **Rundlaufverfahren** (engl. **Round-Robin**, kurz **RR**) das Scheduling für die oben gegebenen vier Fäden für die ersten $22\ \mathrm{ms}$ zu berechnen. Die Zeitscheibenlänge beträgt $\tau = 3\ \mathrm{ms}$ Dazu sei das gegebene Gantt-Diagramm für die Zeitschritte 3 bis 22 zu vervollständigen. Die Zeitschritte 0 bis 2 sind bereits vorausgefüllt:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
A	C	C	C																				
B	▪	▪	▪																				
C	▪	▪	▪																				
D	▪	▪	▪																				

Ein CPU-Stoß soll durch ein **C** und der E/A-Stoß mit einem **E** gekennzeichnet werden. Ein Feld bleibt leer, wenn der Prozess weder E/A- noch CPU-Stoß ausführt

② Gegeben seien zwei Fäden die jeweils eine von zwei Funktionen ausführen (siehe unten) sowie die Semaphore **l** und **n**. Einer der beiden Fäden legt produzierte Daten in einem FIFO-Speicher ***x** ab. Durch den jeweils anderen Faden werden Elemente aus ***x** entfernt und konsumiert. Der Programmtext der Funktionen sei mit den Anweisungen `P(&n)`, `P(&l)` bzw. `V(&n)`, `V(&l)` derart zu vervollständigen, dass bei Ausführung sich Schreib- und Lesevorgang nicht gegenseitig behindern und der Konsument nicht von einem leeren FIFO-Speicher ließt. Deadlocks sind unbedingt zu vermeiden!

Produzent

```
l = 1  // lock (open)
n = 0  // available
```

```
void producer(void* x) {
   ⋮
   while (1) {
      [          ];
      elem* e = produce();
      enqueue(x, e);
      [          ];
      [          ];
   }
}
```

Konsument

```
void consumer(void* x) {
   ⋮
   while (1) {
      [          ];
      [          ];
      elem* e = dequeue(x);
      [          ];
      consume(e);
   }
}
```

③ Durch Markieren (immer nur eine Markierung pro Zeile und Spalte) sei immer einer der vier Verklemmungsbedingungen einer passenden Beschreibung zuzuordnen:

Damit ein Deadlock besteht müssen alle vier Bedingungen erfüllt werden. Durch das Aufheben einer dieser vier Bedingungen werden Deadlocks beseitigt oder vermieden.	Mutual exclusion	Hold and wait	No preemption	Circular wait
Betriebsmittel sind nur unteilbar nutzbar	○	○	○	○
Betriebsmittel sind nicht zurückforderbar	○	○	○	○
Betriebsmittel können nur schrittweise belegt werden	○	○	○	○
Geschlossener Kreis wartender Fäden	○	○	○	○