

Aufgabe 1 (Multiple Choice)**(1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1) = 10 Punkte**

Beantworten Sie folgende Fragen durch Ankreuzen der richtigen Antwort. Für jede falsche Antwort wird ein Punkt abgezogen (es werden minimal 0 Punkte vergeben). Welche der folgenden Aussagen gelten?

- a) Nach der Ausführung einer I/O-Operation informiert der Hardware-Controller über Interrupts *direkt* den Benutzerprozess über die abgeschlossene Operation. Die CPU wird hierbei nicht benötigt.

ja nein

- b) Der Zugriff auf den Hauptspeicher eines Rechners ist etwa 5 Mal langsamer als der Zugriff auf den CPU-Cache.

ja nein

- c) Alle Threads innerhalb eines Prozesses teilen sich dessen Adressraum, lediglich die Deskriptoren für die geöffneten I/O-Geräte werden über Shared-Memory-Bereiche geschützt.

ja nein

- d) Ein MMU ("Memory Management Unit") setzt physikalische in logische Adressen um (z.B. durch Aufaddieren der physikalischen Adresse auf das Base Register des Prozesses).

ja nein

- e) Beim Demand-Paging wird der Seitenaustausch nur im Falle eines Seitenfehlers durchgeführt.

ja nein

- f) Der Peterson-Algorithmus zum wechselseitigen Ausschluss erlaubt nur dann zwei Prozesse im kritischen Bereich, wenn einer der Prozesse erheblich langsamer ist.

ja nein

- g) Sowohl symbolische Verweise als auch benannte Pipes werden in UNIX in einer Inode-Struktur verwaltet.

ja nein

- h) Das Hauptproblem bei der CPU-Scheduling-Strategie LIFO ist die Benachteiligung kurzlaufender Jobs durch Langläufer.

ja nein

- i) Wie beim Paging werden auch beim Swapping einzelne Seiten ausgelagert, jedoch in viel größeren und zusammenhängenden Blöcken.

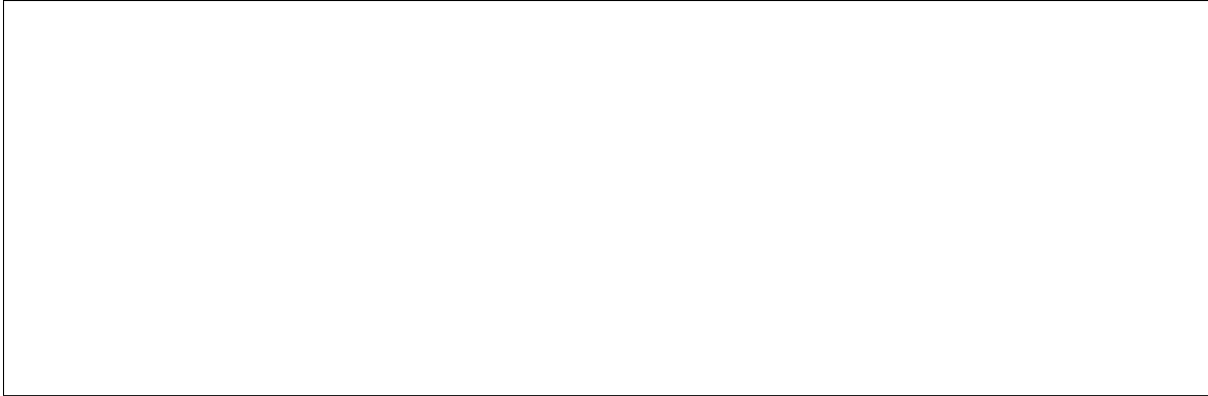
ja nein

- j) Neue Kindprozesse in UNIX müssen sich die CPU-Zeit ihres Vaterprozesses teilen und werden somit benachteiligt. Es empfiehlt sich daher eher, mehrere Instanzen des gleichen Prozesses parallel zu starten.

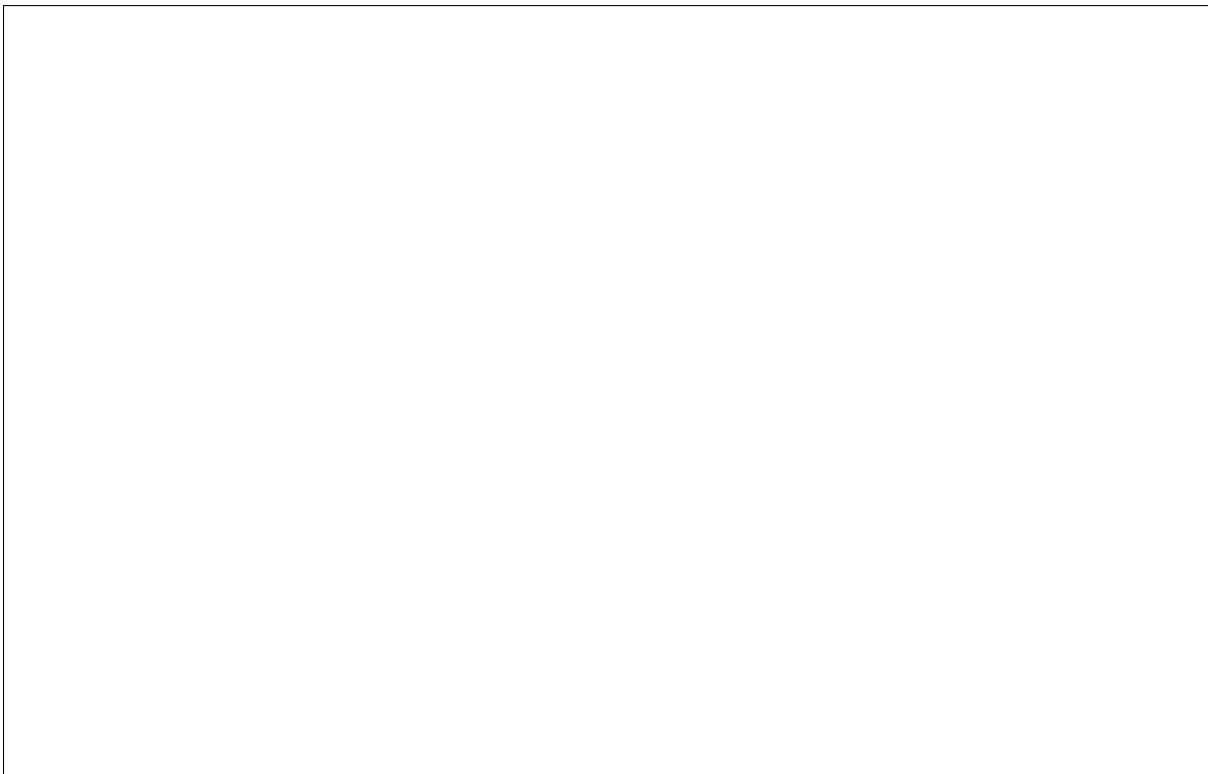
ja nein

Aufgabe 2 (Prozesse)**(3 + 6 + 1) = 10 Punkte**

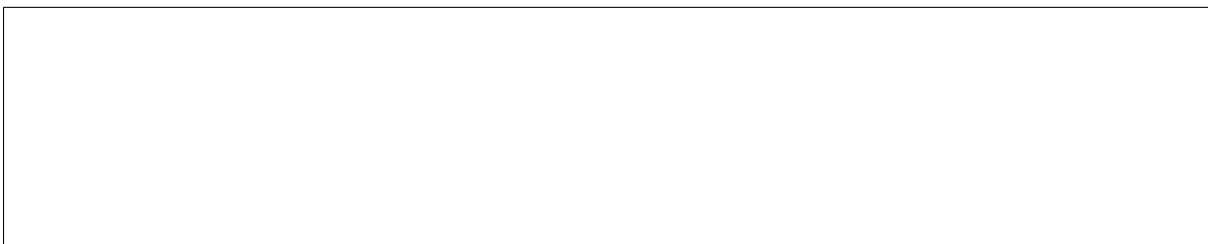
- a) Was versteht man unter einem *Prozesskontrollblock* und welche Informationen enthält er? Nennen Sie 4 Beispiele.



- b) Ein bereits laufender Benutzerprozess möchte 200 MB Daten von einer DMA-fähigen Festplatte auslesen. Was wird seitens des Benutzerprozesses für die Kommunikation mit dem Betriebssystem benötigt? *Skizzieren* Sie den Auslesevorgang in einem Zeit-Diagramm. *Beschriften* Sie die Prozesszustände und die zugehörigen Übergänge.



- c) Welchen *Vorteil* bringt die DMA-Technologie im Beispielszenario b)?



Aufgabe 3 (Prozesse und Prozesszustände)**(2 + 4 + 4 + 2) = 12 Punkte**

- a) Worin liegen die Gemeinsamkeiten und Unterschiede von Prozessen und Threads? Nennen Sie *zwei Gemeinsamkeiten und zwei Unterschiede*.

- b) Was geschieht bei einem Prozesswechsel auf der CPU? *Skizzieren Sie knapp die Aktionen*, die das Betriebssystem unternimmt, um zwischen zwei Prozessen umzuschalten. Begründen Sie auf dieser Basis: sollten *Prozesswechsel bei Verwendung eines Round-Robin-Schedulings sehr häufig oder eher selten* stattfinden?

- c) Geben Sie ein Beispiel mit zwei Prozessen und einem Betriebsmittel an, das zeigt, wie *ein Prozess den Zustand **blocked** erreicht*. Geben Sie dazu in der ersten Tabelle die Ankunftszeitpunkte und Dauern der Prozesse P_1 und P_2 an, sowie nach wie vielen Zeiteinheiten im Zustand **running** auf das Betriebsmittel zugegriffen werden soll.

Tragen Sie in die zweite Tabelle die Zustände der Prozesse für Ihr Beispiel ein. Die **ready**-Warteschlange des Systems wird nach dem FIFO-Verfahren abgearbeitet.

Prozesse:

Prozess	P_1	P_2
Startzeitpunkt		
Benötigte Zeiteinheiten im Zustand running		
Nutzung des BM nach Zeiteinheit		
Benötigte Zeiteinheiten der BM-Nutzung		

Prozesszustände:

	0	1	2	3	4	5	6	7	8	9
P_1										
P_2										

- d) Sie verwenden ein Betriebssystem, das über keinen gemeinsamen Speicher (Shared Memory) verfügt. Allerdings ist es für Ihre Prozessen nötig, miteinander zu kommunizieren. *Wie können Sie eine Interprozesskommunikation erreichen?* Skizzieren Sie das vorgeschlagene Verfahren.

Aufgabe 4 (Prozessverwaltung / C-Programmierung)**(6 + 4 + 1) = 11 Punkte**

- a) *Schreiben* Sie ein C-Programm, das in einer Linux/Unix-Umgebung einen Kindprozess erzeugt. Der Vaterprozess soll in einer Endlosschleife "Sys" ausgeben. Der Kindprozess soll ebenfalls in einer Endlosschleife "Pro" ausgeben. Verwenden Sie dazu das folgende Rahmenprogramm:

```
int main() {  
  
    if (          ) {  
  
    } else {  
  
    }  
  
    return -1;  
}
```

- b) Ändern Sie den Code des Vaterprozesses: Sobald er 10 Mal "Sys" ausgegeben hat, soll er seinen Kindprozess beenden. Verwenden Sie dazu die Funktion `kill(pid_t pid, int sig)`.

- c) *Beschreiben* Sie informell in 1-2 Sätzen, wie man den obigen Code erweitern muss, damit der Kindprozess die Signale seines Vaterprozesses (gleichzeitig auch Signale anderer Prozesse) ignoriert und seine Ausgabe fortsetzt.

Aufgabe 5 (Scheduling)**(4 + 4 + 4 + 2) = 14 Punkte**

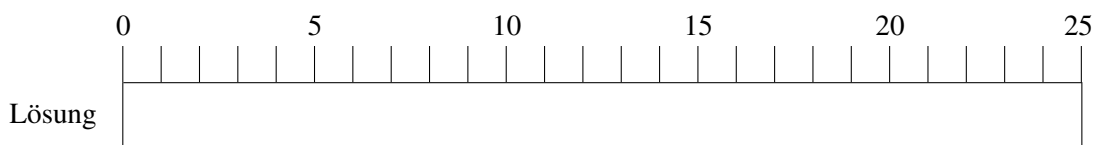
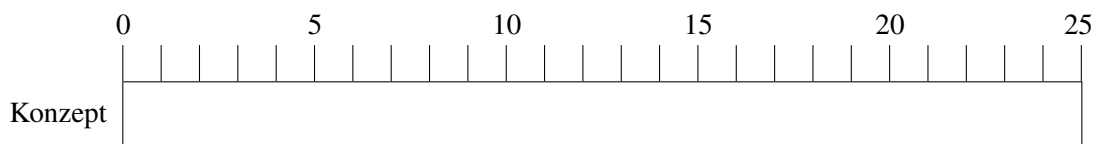
Gegeben sei ein Rechnersystem mit einer CPU und 5 Prozessen P_0, \dots, P_4 . In der folgenden Tabelle ist für diese Prozesse angegeben, zu welchen Zeitpunkten sie das System betreten sowie für wie viele Zeiteinheiten sie die CPU benötigen:

Prozess	Ankunftszeitpunkt	Dauer
P_0	0	5
P_1	2	2
P_2	3	6
P_3	6	7
P_4	9	2

Ein Ankunftszeitpunkt von t bedeutet, dass der Prozess zu diesem Zeitpunkt im Zustand **ready** auf Zuteilung der CPU wartet. Zwischen zwei aufeinanderfolgenden Zeitpunkten t und $t + 1$ vergeht eine Zeiteinheit. Das Umschalten zwischen zwei Prozessen nehme keine Zeit in Anspruch. Hat ein Prozess für die unter **Dauer** angegebenen Zeiteinheiten die CPU belegt, verlässt er sofort das System, ohne weitere CPU-Zeit zu beanspruchen.

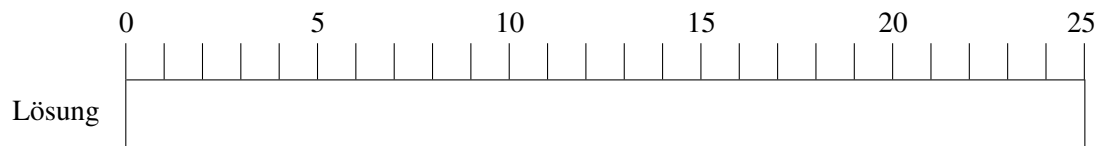
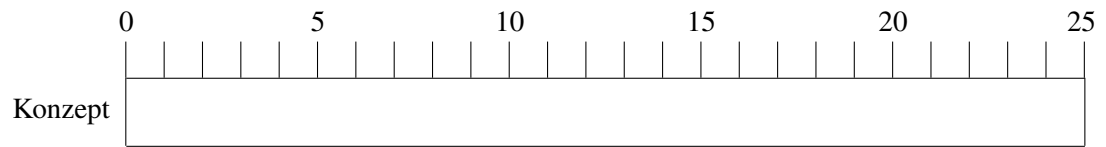
Im Folgenden sind 3 Scheduling-Strategien angegeben. *Illustrieren Sie die Belegung der CPU anhand der vorgegebenen Gantt-Charts.* Die Zahlen über den jeweiligen Charts bezeichnen Zeitpunkte und sind lediglich vorgegeben, um eine bessere Orientierung bei der Erstellung der Lösung zu geben. Zu jeder Strategie sind zwei Vorlagen vorhanden; die mit „Konzept“ beschriftete Vorlage können Sie zur Erarbeitung der Lösung verwenden. **Wenn sowohl die mit „Konzept“ als auch die mit „Lösung“ beschriftete Vorlage Einträge enthalten, wird nur die mit „Lösung“ beschriftete Vorlage gewertet!**

Berechnen Sie außerdem für jede Strategie die *mittlere Wartezeit* für Ihren Schedule. Führen Sie Ihre Berechnungen in dem mit „Rechnung“ bezeichneten Feld durch und tragen Sie das Endergebnis in das mit „Ergebnis“ bezeichnete Feld ein.

a) Strategie: **LIFO**Mittlere Wartezeit:

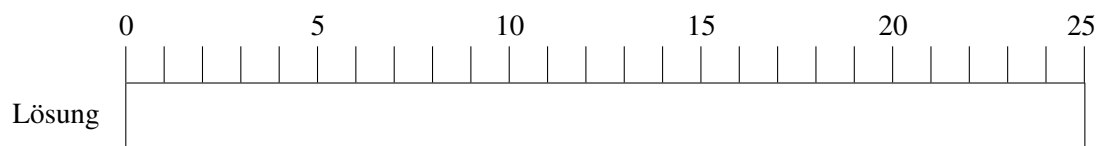
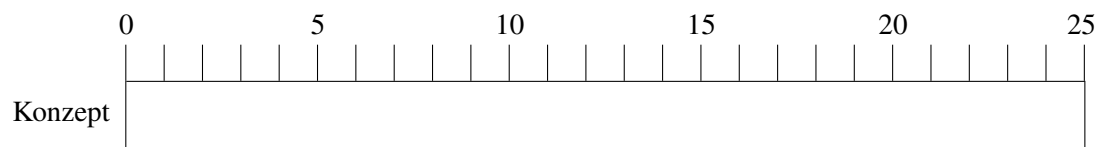
Rechnung:

Ergebnis:

b) Strategie: **SRPT**Mittlere Wartezeit:

Rechnung:

Ergebnis:

c) Strategie: **Round Robin mit Quantum $Q = 4$** Mittlere Wartezeit:

Rechnung:

Ergebnis:

d) Sind die Strategien wie hier verwendet *work conserving*? Begründen Sie Ihre Antwort!

Aufgabe 6 (Hauptspeicherverwaltung)

(1 + 9 + 2 + 3) = 15 Punkte

- a) Beschreiben Sie in einem Satz, wofür in Betriebssystemen die *Lifetime-Funktion* verwendet wird.

--

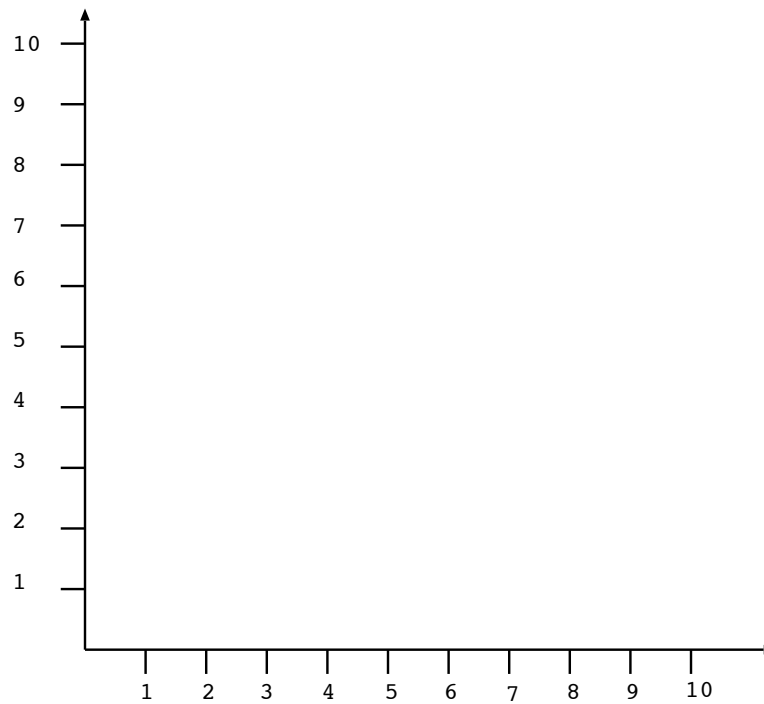
- b) Zeichnen Sie in die Vorlage auf der nächsten Seite den *Graphen der Lifetime-Funktion* $L(m)$ für $m = 1 \dots 10$ und geben Sie zusätzlich in der Tabelle die *Zahl der Seitenfehler* für $m = 1 \dots 10$ an zu einem Programm an, das durch den Referenzstring ω mit

$$\omega = 0\ 5\ 1\ 4\ 2\ 5\ 3\ 0\ 1\ 4\ 1\ 2\ 5\ 4\ 2\ 5\ 4\ 2\ 5\ 1\ 0\ 2\ 1\ 5$$

gekennzeichnet ist. Die Zeit, die zwischen zwei Seitenanfragen vergeht, soll jeweils eine Zeiteinheit betragen. Die Seitenersetzung erfolge mittels der folgenden Strategie: muss eine neue Seite eingefügt werden, so wird diejenige verdrängt, auf die am längsten nicht mehr zugegriffen wurde. Gehen Sie davon aus, dass das Working Set zu Beginn mit anderen als den angegebenen Seiten gefüllt ist.

$\omega =$	0	5	1	4	2	5	3	0	1	4	1	2	5	4	2	5	4	2	5	1	0	2	1	5	
m = 2																									
Seitenfehler																									
Seitenrahmen																									
Seitenrahmen																									
m = 3																									
Seitenfehler																									
Seitenrahmen 1																									
Seitenrahmen 2																									
Seitenrahmen 3																									
m = 4																									
Seitenfehler																									
Seitenrahmen 1																									
Seitenrahmen 2																									
Seitenrahmen 3																									
Seitenrahmen 4																									
m = 5																									
Seitenfehler																									
Seitenrahmen 1																									
Seitenrahmen 2																									
Seitenrahmen 3																									
Seitenrahmen 4																									
Seitenrahmen 5																									
m = 6...10																									
Seitenfehler																									
Seitenrahmen 1																									
Seitenrahmen 2																									
Seitenrahmen 3																									
Seitenrahmen 4																									
Seitenrahmen 5																									
Seitenrahmen 6																									

m	Seitenfehler	L(m) Werte
1		
2		
3		
4		
5		
6...10		



- c) Welche *optimale Einstellung der SpeichergroÙe* ergibt sich bei Anwendung des *primären Knie-Kriteriums*?
Zeichnen Sie zusätzlich die *zugehörige Gerade* in Ihren Graphen ein.

- d) Geben Sie *alle Working-Sets* zum Zeitpunkt $t = 20$ an.

Aufgabe 7 (Wechselseitiger Ausschluss)**(5 + 5 + 5) = 15 Punkte**

Zur korrekten Lösung des Problems des wechselseitigen Ausschlusses müssen drei Bedingungen erfüllt werden. Ein angehender Programmierer hat eine Lösung für zwei Prozesse P_i und P_j entworfen. Im Folgenden ist die Lösung für Prozess P_i gegeben (die Lösung für Prozess P_j sieht analog aus, nur mit vertauschten Indizes i und j):

```
repeat
  flag[i] := TRUE;
  turn := j;
  while (flag[i] and turn = j) do noop;
  kritischer Bereich;
  flag[i] := FALSE;
  unkritischer Bereich;
until FALSE;
```

Nennen Sie im Folgenden die drei Bedingungen für den wechselseitigen Ausschluss, geben Sie jeweils kurz die Bedeutung der Bedingung an und begründen Sie für jede Bedingung, warum die gegebene Lösung sie erfüllt bzw. nicht erfüllt.

a) Bedingung 1:

Bedeutung:

Ist die Bedingung erfüllt?

ja
nein

Begründung:

b) Bedingung 2:

Bedeutung:

Ist die Bedingung erfüllt? ja nein

Begründung:

c) Bedingung 3:

Bedeutung:

Ist die Bedingung erfüllt? ja nein

Begründung:

Aufgabe 8 (Semaphore)**(2 + 3 + 4 + 10) = 19 Punkte**

Sie sollen eine Zufahrtsregelung für eine altersschwache Brücke entwerfen. Aus Sicherheitsgründen darf sich immer nur eine bestimmte Anzahl von Fahrzeugen gleichzeitig auf der Brücke befinden. Sie wollen Semaphore mit assoziierten Warteschlangen verwenden, um die Zufahrt zur Brücke zu regulieren und haben bereits die folgenden Vorgaben:

```
Fahrzeug(boolean Richtung)
{
    fahre_auf_Brücke_zu();
    enterBridge();
    überquereBrücke(); // kritischer Bereich
    exitBridge();
    fahre_weiter();
}
```

Die Variable *Richtung* wird pro Fahrzeug gesetzt und gibt an, in welche Richtung ein Fahrzeug die Brücke überqueren will.

Geben Sie die Implementierung von *enterBridge* und *exitBridge* sowie die *globalen Initialisierungsbedingungen der verwendeten Semaphore* (und, falls verwendet, globalen Variablen) für die folgenden vier Fälle an.

- a) Zu jedem Zeitpunkt darf sich nur *ein einziges Fahrzeug auf der gesamten Brücke* befinden.

- b) Zu jedem Zeitpunkt darf sich *pro Fahrtrichtung ein Fahrzeug auf der Brücke* befinden.

- c) Fahrzeuge der beiden Fahrrichtungen dürfen nur *abwechselnd* die Brücke überqueren. Sie können frei entscheiden, welche der Fahrrichtungen beginnen darf.



- d) Zu jedem Zeitpunkt darf nur *eine Fahrrichtung* genutzt werden, aber es dürfen *beliebig viele Fahrzeuge* in diese Richtung fahren.



Aufgabe 9 (Deadlocks)**(8 + 5 + 1) = 14 Punkte**

Gegeben seien vier Prozesse P_1 , P_2 , P_3 und P_4 , die zur Lösung Ihrer Aufgaben eine bestimmte Anzahl der Betriebsmittel R_1 und R_2 benötigen. Von beiden Betriebsmitteln gibt es jeweils 7 Exemplare.

Der Prozess P_i benötigt während seiner Laufzeit zu jedem Zeitpunkt maximal $Max(P_i)$ Einheiten der Betriebsmittel R_1, R_2 :

$$Max(P_1) = (7, 4), \quad Max(P_2) = (4, 4), \quad Max(P_3) = (1, 3), \quad Max(P_4) = (1, 3)$$

Die aktuelle Betriebsmittelzuteilung sei wie folgt gegeben:

$$H_1 = (3, 2), \quad H_2 = (2, 2), \quad H_3 = (1, 0), \quad H_4 = (0, 1)$$

- a) Überprüfen Sie mit Hilfe des *Banker-Algorithmus*', ob sich das System in einem sicheren Zustand befindet.

Betrachten Sie nun ein System mit fünf Prozessen P_1, \dots, P_5 und fünf Betriebsmitteln R_A, \dots, R_E , von denen jeweils nur ein Exemplar zur Verfügung steht und die von nur einem Prozess gleichzeitig verwendet werden können. Die Anforderungen von Betriebsmitteln durch die Prozesse ist in der folgenden Tabelle gegeben.

Zeit	P_1	P_2	P_3	P_4	P_5
1			(A;2)		
2	(C;5)			(D;7)	
3		(E;4)		(C;3)	
4		(D;4)			(A;3)
5			(B;4)		(E;4)
6	(B;3)		(A;3)		
7					

Ein Eintrag der Form $(X; a)$ in der Spalte P_i bedeutet, dass das Betriebsmittel R_X für a Zeiteinheiten vom Prozess P_i angefordert wird. Ein Prozess kann seine Arbeit erst dann fortsetzen, wenn ihm alle angeforderten Betriebsmittel zugewiesen wurden.

Sind einem Prozess die angeforderten Betriebsmittel zum Zeitpunkt t zugewiesen worden, werden sie in der Zeit von $t + 1$ bis $t + a$ verwendet und stehen zum Zeitpunkt $t + a + 1$ wieder zur Verfügung.

b) Verdeutlichen Sie die Abhängigkeiten der Prozesse zum Zeitpunkt 7 anhand eines *Wait-For-Graphen*.

c) Liegt in dieser Situation ein *Deadlock* vor?

ja nein