JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN
Fachbereich Informatik & Mathematik (12)
AG Adaptive Systemarchitektur

Prof. Dr. R. Brause
Robert-Mayer-Str. 11-15
Tel.: +049-(0)69-798-23977
Fax: +049-(0)69-798-21228
Email: RBrause@cs.uni-frankfurt.de

Betriebssysteme 1 WS05/06

Musterlösung der Klausur

Vorname:	
Nachname:	
Matrikelnummer:	
Geburtsdatum:	
Studiengang:	

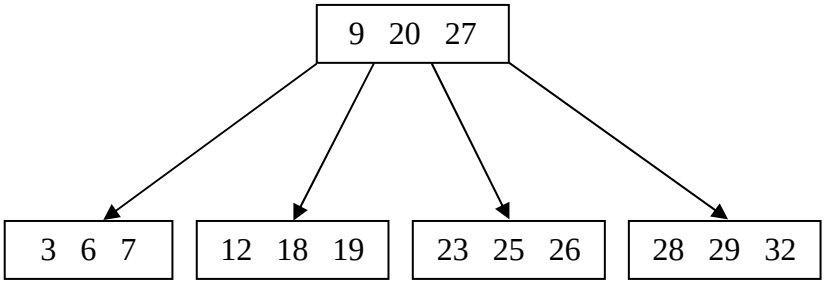
1	2	3	4	5	6	7	8	Summe

Name:	Matrikelnummer:

Name:	Matrikelnummer:

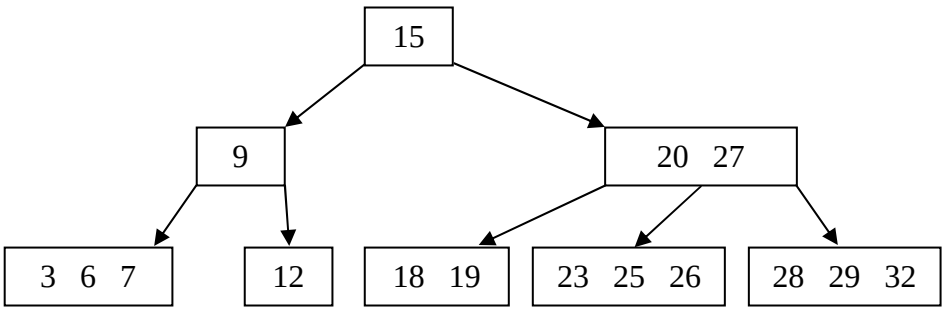
Aufgabe 1 (B-Bäume, B*-Bäume) (35 Punkte)

a) *Es sei **m = 4**. Fügen Sie die Schlüssel **15**, **40** und **24** nacheinander in den folgenden **B-Baum** ein:*

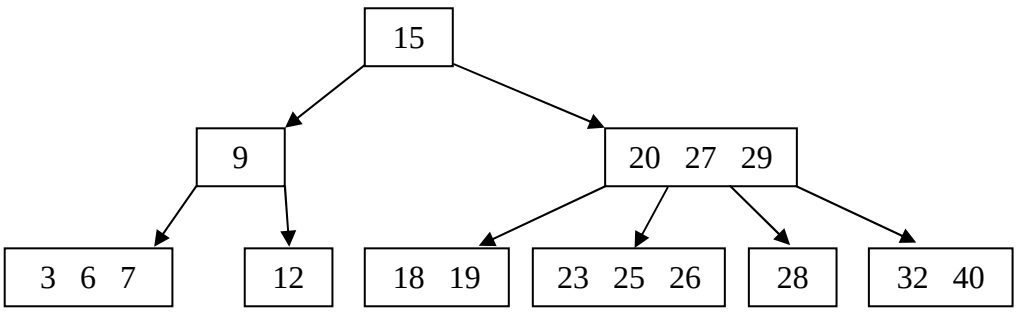


Zeichnen Sie den aktuellen Baum nach jedem Einfügen. Es sollen hierbei keine Verschiebe-Operationen durchgeführt werden.

Einfügen von **Schlüssel 15**: (2 Punkte)

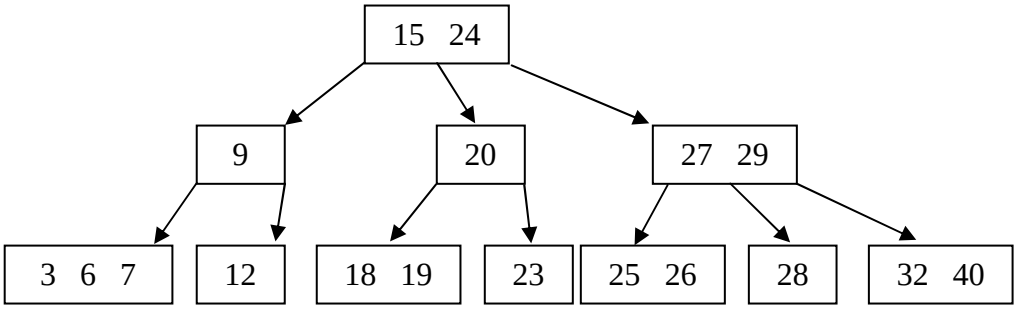


Einfügen von **Schlüssel 40**: (2 Punkte)

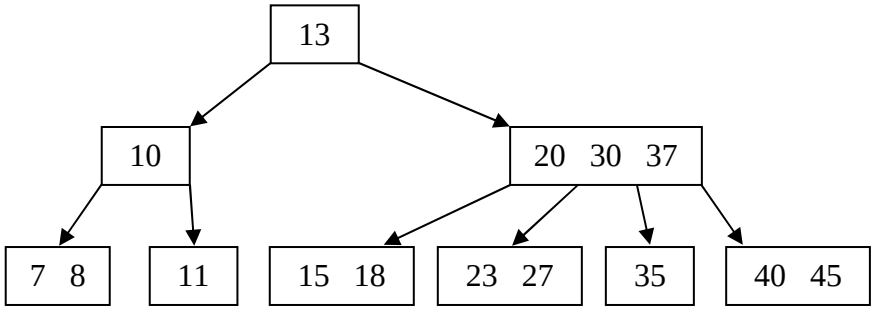


Name:	Matrikelnummer:

Einfügen von **Schlüssel 24**: (2 Punkte)

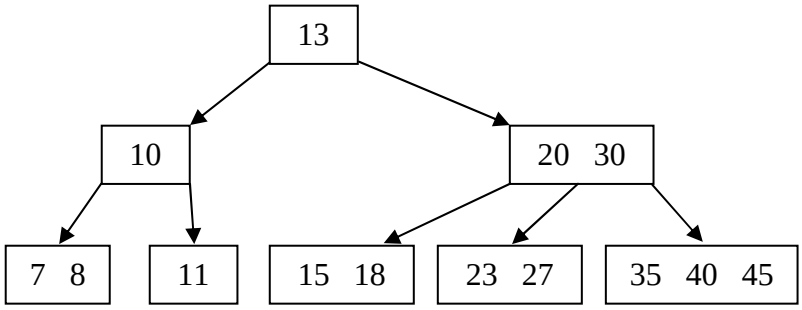


b) *Es sei **m=4**. Löschen Sie die Schlüssel **37** und **10** nacheinander aus dem folgenden **B-Baum**:*

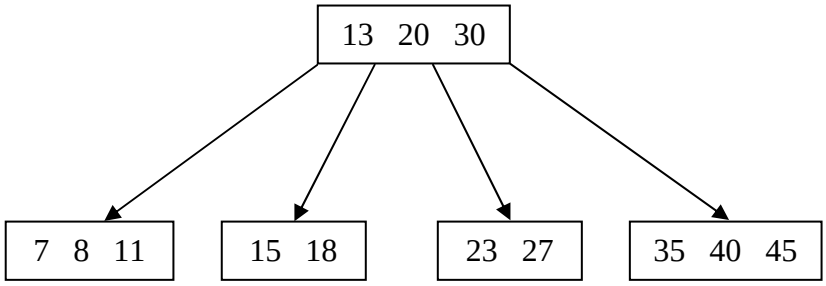


Achten Sie darauf, dass die Löschoperation der inversen Einfügeoperation entsprechen muss. Zeichnen Sie den aktuellen Baum nach jeder Löschoperation.

Löschen von **Schlüssel 37**: (3 Punkte)



Löschen von **Schlüssel 10**: (3 Punkte)



c) *Wie viele Schlüssel enthält die Wurzel eines **B*-Baumes** maximal?* *(1 Punkte)*

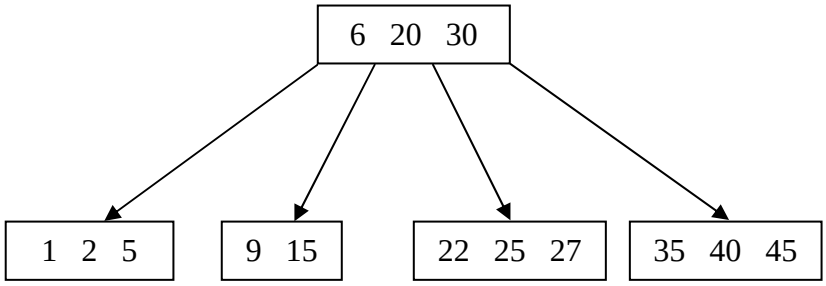
$2\left\lfloor \frac{2m-2}{3} \right\rfloor$ Schlüssel

Welchem Zweck dient diese Regel? *(2 Punkte)*

Durch diese Regel kann die Wurzel in zwei Behälter der Größe $\left\lfloor \frac{2m-2}{3} \right\rfloor$ aufgeteilt werden. Diese Regel ist also notwendig um die B*-Baum Bedingung, dass ein Knoten immer zu $\left\lfloor \frac{2m-2}{3} \right\rfloor$ gefüllt sein muss, zu erfüllen.

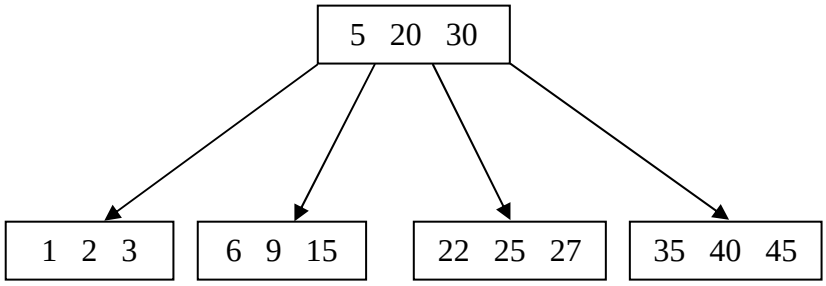
Name:	Matrikelnummer:

d) *Es sei **m = 4**. Fügen Sie die Schlüssel **3**, **50** und **4** nacheinander in den folgenden **B*-Baum** ein:*

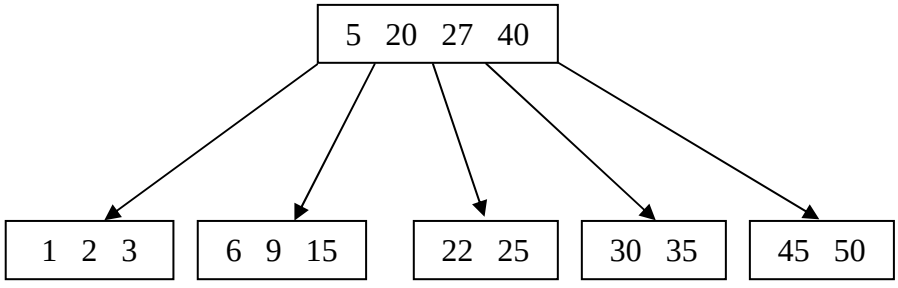


Zeichnen Sie den aktuellen Baum nach jedem Einfügen.

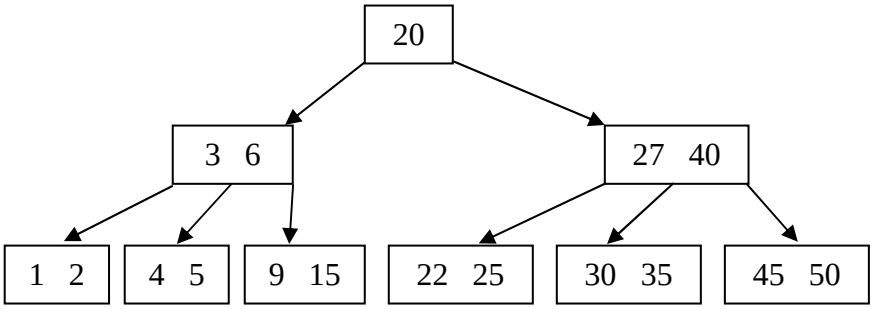
Einfügen von **Schlüssel 3**: (3 Punkte)



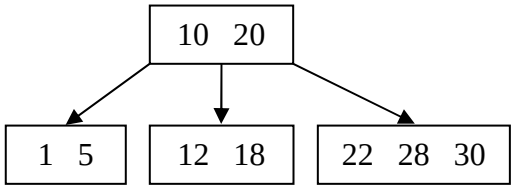
Einfügen von **Schlüssel 50**: (3 Punkte)



Einfügen von **Schlüssel 4**: (3 Punkte)

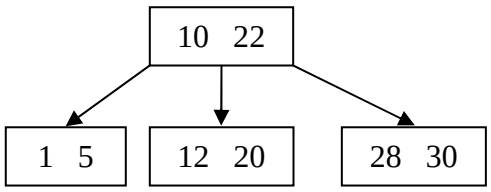


e) *Es sei **m =4**. Löschen Sie die Schlüssel **18**, **20** und **12** nacheinander aus dem folgenden **B*-Baum**:*



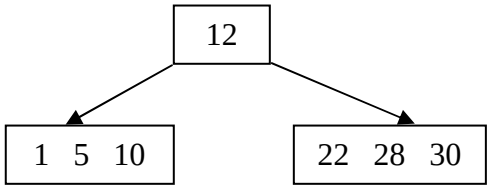
Achten Sie darauf, dass die Löschoperation der inversen Einfügeoperation entsprechen muss. Zeichnen Sie den aktuellen B-Baum nach jeder Löschoperation.*

Löschen von **Schlüssel 18**: (3 Punkte)

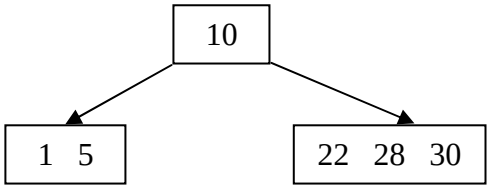


Name:	Matrikelnummer:

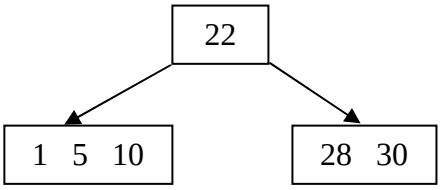
Löschen von **Schlüssel 20**: (3 Punkte)



Löschen von **Schlüssel 12**: (3 Punkte)



oder



f) *Wäre die Operation* Verschieben zwischen Schlüsselbehältern *theoretisch auch bei einem B-Baum möglich? Begründen Sie Ihre Antwort?* *(2 Punkte)*

Die Operation *Verschieben* wäre auch bei einem B-Baum möglich, da sie die Bedingung eines B-Baumes nicht verletzt. Sie würde sogar eine Verbesserung der Verteilung der Schlüssel bewirken, ist aber wie gesagt nicht erforderlich um die Bedingungen zu erfüllen.

Name:	Matrikelnummer:

Aufgabe 2 (Scheduling) (25 Punkte)

a) *Gegeben sind die folgenden Prozesse:*

Prozess:	A	B	C	D	E	F
Priorität:	10	15	7	12	1	3
Bedienzeit:	4	5	3	2	6	1

Ordnen Sie die Prozesse in der Warteschlange nach den folgenden Verfahren und geben Sie zu jedem die mittlere Ausführungszeit an. Wir nehmen an, dass dabei keine Aus- und Eingabe stattfinden muss.

- ***Highest Ratio Next*** *(im ersten Schritt den Kürzesten)* *(6 Punkte)*

  Schritt 1: Es wird **F** ausgewählt, da kürzeste. **F** beendet nach 1 Zeitschritt.
  Antwortzeit des Prozesses = Bearbeitungszeit des Prozesses + 1

  Schritt 2: Berechnen der Quotienten Antwortzeit/Bearbeitungszeit. Feststellung: Prozess **D** hat den größten, also wird **D** ausgeführt. **D** beendet nach 3 Zeitschritten.
  Antwortzeit des Prozesses = Bearbeitungszeit des Prozesses + 2

  Schritt 3: wie im Schritt 2. Prozess **C** wird gewählt. **C** ist beendet nach 6 Zeitschritten.
  Antwortzeit des Prozesses = Bearbeitungszeit des Prozesses + 3

  Schritt 4: Prozess **A** wird gewählt. **A** ist beendet nach 10 Zeitschritten.
  Antwortzeit des Prozesses = Bearbeitungszeit des Prozesses + 4

  Schritt 5: Prozess **B** wird gewählt. **B** ist beendet nach 15 Zeitschritten.
  Antwortzeit des Prozesses = Bearbeitungszeit des Prozesses + 5

  Schritt 6: Prozess **E** wird gewählt. **E** ist beendet nach 21 Zeitschritten.

  Rechnung: 1 + 3 + 6 + 10 + 15 + 21 = 56 => 56/6 = 9.33 Zeitschritte im Schnitt.

- ***Round Robin*** *(5 Punkte)*
  Gehen Sie dabei von einer unendlich kleinen Zeitscheibe aus, so dass die Reihenfolge der Prozesse unerheblich wird, das heißt, sie laufen quasi parallel ab.

  F endet als erster nach: 6*1=6 Zeitschritten
  D endet nach: 6+5*1=11 Zeitschritten
  C endet nach: 11+4*1=15 Zeitschritten
  A endet nach: 15+3*1=18 Zeitschritten
  B endet nach: 18+2*1=20 Zeitschritten
  E endet nach: 20+1*1=21 Zeitschritten

  Rechnung: 6 + 11 + 15 + 18 + 20 + 21 = 91 => 91/6 = 15,17 Zeitschritte im Schnitt

- ***Shortest Job First*** *(3 Punkte)*

  Ordnen nach Berechnungszeiten ergibt: **FDCABE**

  Rechnung: 1 + 3 + 6 + 10 + 15 + 21 = 56 => 56/6 = 9.33 Zeitschritte im Schnitt.

b) *Welche der obigen Scheduling-Verfahren können zu einem Verhungern der Prozesse führen und welche nicht? Begründen Sie ihre Antwort.* *(3 Punkte)*

Shortest Job First kann zu Verhungern führen, da es passieren kann, dass Prozesse mit sehr großer Bearbeitungszeit nie aus der Warteschlange herauskommen, da sich immer wieder neue kürzere Prozesse vor ihm einordnen.

Round Robin kann nicht zu Verhungern führen, da jeder Prozess einen gleichen Anteil am Prozessor bekommt. (Zeitscheibenprinzip).

Highest Ratio Next kann nicht zum Verhungern führen, da im Gegensatz zu Shortest Job First, der Quotient von Prozessen (mit langer Bearbeitungsdauer) mit wachsender Wartezeit immer größer wird.

c) *Welche Anforderungen werden an **Echtzeit-Scheduling-Algorithmen** gestellt?* *(2 Punkte)*

Die Prozesse müssen bis zu einem gegebenem Zeitpunkt (Deadline) ausgeführt werden.

Dazu muss ein Echzeit-Scheduling-Algorithmus die Deadline, die Ausführungsrate, die Dauer und die Betriebsmittelbelegung der Prozesse berücksichtigen. Bei einem „Hart-Echtzeitsystem“ zum Beispiel müssen die Deadlines strikt eingehalten werden, da eine Verspätung katastrophale Folgen hätte.

d) *Erläutern Sie die Echtzeit-Scheduling-Algorithmen* Minimal-Deadline-First *und* Rate-Monotonic-Scheduling. *(3 Punkte)*

Minimal-Deadline-First: Bei diesem Verfahren wird der Prozess ausgeführt, der zu einem gegebenem Zeitpunkt an nahesten an seiner Deadline ist.

Rate-Monotonic-Scheduling: Dieses Verfahren ordnet den Prozessen mit einer hohe Ausführungsrate eine hohe Priorität zu, d.h wenn man sich zwischen mehreren Prozessen entscheiden muss wird der mit der höheren Ausführungsrate ausgeführt.

g) *Beschreiben Sie, welche Probleme bei diesen beiden Algorithmen auftreten können. Wie könnte man diese Probleme vermeiden?* *(3 Punkte)*

Da bei RMS Bedingungen (wie z.B. die Dauer des Prozesses) nicht beachtet werden, kann es dazu kommen dass Prozesse mit niedriger Ausführungsrate eine Prioritätserhöhung (Prioritätsinversion) erhalten müssen.
MDF kann dazu führen das Prozesse mit einer geringen Deadline und geringer Prio, aber hoher Bearbeitungszeit bevorzugt werden und somit alle anderen wartenden Prozesse ihre Deadline verletzen.
Dies könnte man verhindert indem man eine zusammengesetzte Auswahlbedingung wie z.B. Deadline -Bearbeitungsdauer einführt.

Name:	Matrikelnummer:

Aufgabe 3 (Race Conditions) (15 Punkte)

Gegeben ist der Prozess Lichtschranke *der fortlaufend die Lichtschranke einer elektronischen Tür überprüft. Falls die Lichtschranke auslöst, schreibt er den Befehl „open“ in die globale Variable Aktion (bzw. „close“ wenn die Lichtschranke ungestört ist). Der Prozess* Türsteuerung *liest diese Variable und führt die geforderte Aktion aus. Nachstehend ist das Aktionsschema übersichtsweise abgebildet.*

```
global Aktion
```

Lichtschranke

```
LOOP
  IF ausgelöst     THEN Aktion:=open
  IF NOT ausgelöst THEN Aktion:=close
END
```

Türsteuerung

```
LOOP
  IF Aktion=open  THEN Tür.auf()
  IF Aktion=close THEN Tür.zu()
END
```

a) *Kann hierbei eine* race condition *auftreten? Falls ja, erklären Sie wo und wie dies passiert. Falls nein, erläutern Sie warum in diesem Fall keine* race condition *auftreten kann.* *(4 Punkte)*

Es können sogar mehrere *race conditions* auftreten, z.B.:
Der Prozess Türsteuerung hat gerade die Aktion `close` gelesen und wird unterbrochen, in der Zwischenzeit läuft jemand durch die Schranke und der Prozess Lichtschranke setzt Aktion auf `open`. Der Prozess Türsteuerung kommt wieder an die Reihe und fährt fort wo er aufgehört hat, d.h. er schließt die Tür, obwohl er die Tür eigentlich öffnen müsste.

Name:	Matrikelnummer:

b) *Der Programmierer Klaus-Peter H. (Name geändert) hat versucht, die vermeintlichen* race conditions *zu verhindern, indem er Semaphore in den Quellcode eingebaut hat:*

```
global Aktion, s
```

Lichtschranke

```
P(s)
LOOP
  IF ausgelöst     THEN Aktion:=open
  IF NOT ausgelöst THEN Aktion:=close
END
V(s)
```

Türsteuerung

```
P(s)
LOOP
  IF Aktion=open  THEN Tür.auf()
  IF Aktion=close THEN Tür.zu()
END
V(s)
```

Treten jetzt immer noch race conditions *auf, bzw. treten nach wie vor keine auf? Begründen Sie.* *(3 Punkte)*

Es treten keine *race conditions* mehr auf, da der gesamte Prozess durch die Semaphore geschützt wurde.

Erkennen Sie ein Problem, das durch das Setzen der Semaphore aufgetreten ist? *(3 Punkte)*

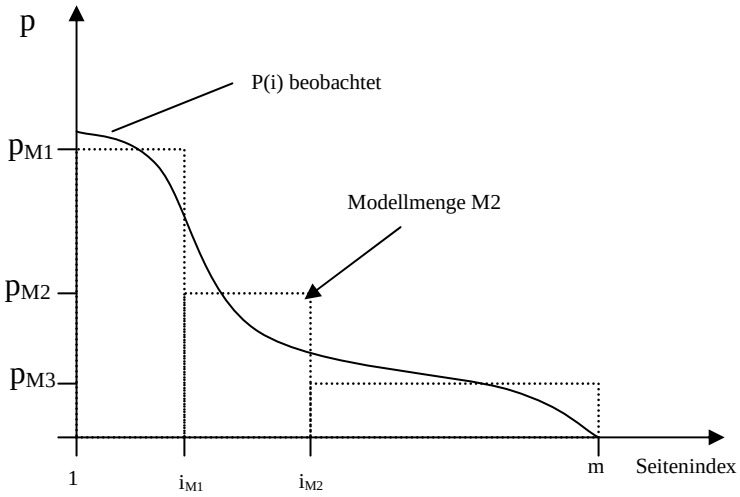
Die Semaphore ist völlig falsch gesetzt, da die Passieren Methode vor der Schleife und die Verlassen Methode nach der Schleife definiert sind. Dies führt dazu, dass der erste Prozess, der das Semaphor passieren darf, ewig läuft und der andere Prozess nie dran kommt.

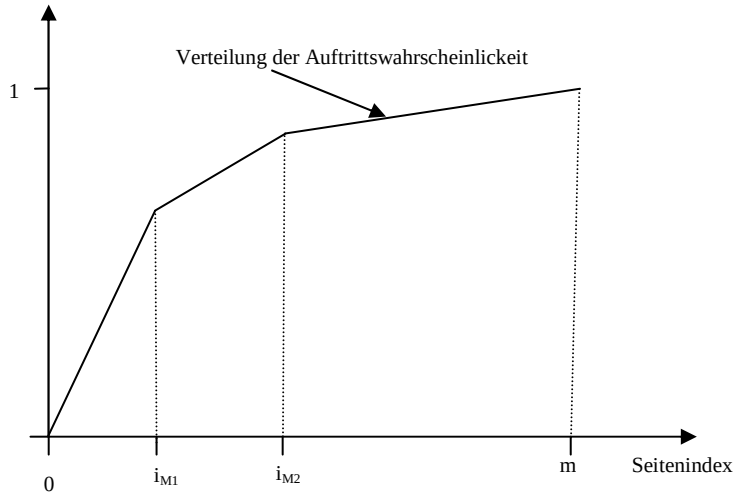
c) *Beschreiben Sie, falls nötig, welche Art von Semaphore Sie verwenden würden und an welcher Stelle Sie diese platziert hätten.* *(5 Punkte)*

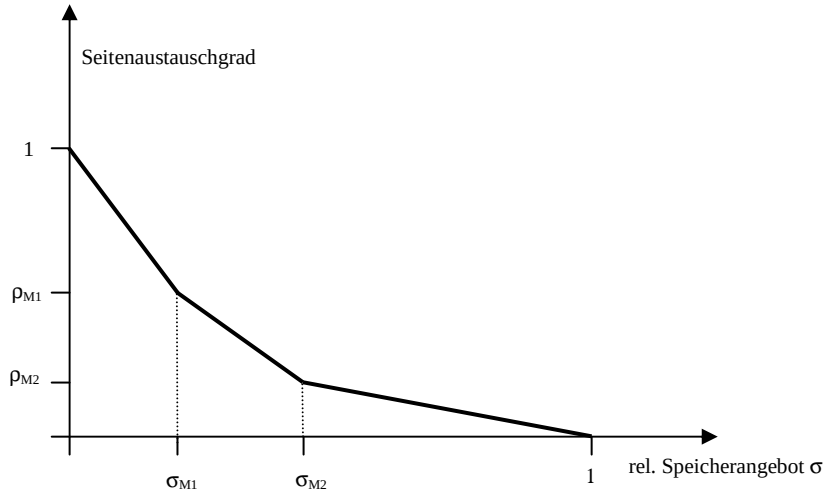
Um alles richtig zu machen, sollte man mutex-Semaphore verwenden, da immer nur ein Prozess im kritischen Abschnitt sein sollte. Die *Passieren*-Operation sollte am Anfang des Schleifeninhalts stehen und die *Verlassen*-Operation am Ende des Schleifeninhalts. So kann man sicher gehen, dass jeder einzelne Schleifendurchlauf geschützt ist.

Name:	Matrikelnummer:

Aufgabe 4 (Thrashing) (20 Punkte)







Name:	Matrikelnummer:

a) *Beschriften Sie die obigen Diagramme (Achsen, Achsenabschnitte, Graphen usw.) und beschreiben Sie, was die Diagramme (im Bezug zu Thrashing) aussagen.* *(10 Punkte)*

Das erste Diagramm stellt die Referenzierungswahrscheinlichkeit der Seiten und die drei Modellierungsmengen dar.
Das zweite Diagramm zeigt die Verteilungsdichtefunktion der modellierten Referenzierungswahrscheinlichkeiten, d.h. die Auftrittswahrscheinlichkeiten der einzelnen Intervalle.
Das dritte Diagramm zeigt die Seitenaustauschgerade in Bezug zu dem relativen Speicherangebot oder der Anzahl der Prozesse im Speicher, je nach Beschriftung der X-Achse.

b) *Angenommen Sie haben ein Thrashing-Problem im obigen Modell. Welche Situationen können bei verschiedenen Werten von σ_w (relatives Speicherangebot bei dem $t_w > t_s$ wird) auftreten? Teilen Sie die entstehenden Situationen in qualitativ unterschiedliche Fälle auf.* *(10 Punkte)*

Fall 1: σ_w fällt in das Intervall $[0,\sigma_{M1}]$, dann entsteht ein linearer Bereich im Intervall $[\sigma_w ,1]$ und ein überproportionaler Bereich im Intervall $[0, \sigma_w]$.

Fall 2: σ_w fällt in das Intervall $[\sigma_{M1}, \sigma_{M2}]$, dann entsteht eine linearer Bereich im Intervall $[\sigma_w ,1]$, ein überproportionaler Bereich im Intervall $[\sigma_{M1} , \sigma_w]$ und ein stark überproportionaler Bereich im Intervall $[0, \sigma_{M1}]$.

Fall 3: σ_w fällt in das Intervall $[\sigma_{M2},1]$, dann entsteht ein linearer Bereich im Intervall $[\sigma_w,1]$, ein überproportionaler Bereich im Intervall $[\sigma_{M2} , \sigma_w]$, ein stark überproportionaler Bereich im Intervall $[\sigma_{M1}, \sigma_{M2}]$ und ein sehr stark überproportionaler Bereich im Intervall $[0,\sigma_{M1}]$.

Name:	Matrikelnummer:

Aufgabe 5 (Multiprozessor-Scheduling) (25 Punkte)

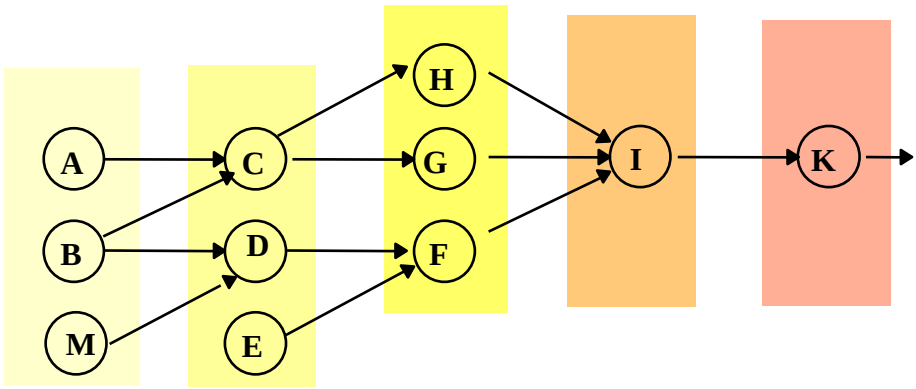
*Gegeben sind die Prozesse **A** bis **J** mit den angegebenen Prioritäten (wobei 5 die höchste ist) und Bearbeitungszeiten:*

Prozess:	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*K*	*M*
Priorität:	*1*	*3*	*3*	*5*	*2*	*4*	*5*	*2*	*3*	*1*	*5*
Bedienzeit:	*2*	*3*	*2*	*4*	*3*	*5*	*2*	*3*	*4*	*3*	*4*

Außerdem gelten die folgenden Präzedenzrelationen:

B >> C >> H >> I >> K
M >> D
A >> C
C >> G >> I
B >> D >> F >> I
E >> F

a) *Erstellen Sie den zu den Präzedenzrelationen gehörenden Präzedenzgraphen.* *(5 Punkte)*



b) *Sie haben drei Prozessoren zur Verfügung. Führen Sie jetzt ein Multiprozessor-Scheduling nach der* Latest-*Methode durch und zeichnen Sie dazu ein Gantt-Diagramm.* *(7 Punkte)*

P1										A			C			H				I				K	
P2				M								E							G						
P3					B						D						F								

Name:	Matrikelnummer:

c) Verwenden Sie jetzt das List-*Scheduling und zeichnen Sie das entstehende Diagramm. (7 Punkte)*

P1	M			D			F		I	K	
P2	B		A		C	G					
P3	E					H					

d) Gegeben ist der folgenden Pseudocode:

```
PROCESS A: BEGIN;                      Taskbody(A); V(C1);          END;
PROCESS B: BEGIN;                      Taskbody(B); V(C2); V(D1);   END;
PROCESS C: BEGIN; P(C1); P(C2);        Taskbody(C); V(G);  V(H);    END;
PROCESS D: BEGIN; P(D1); P(D2);        Taskbody(D); V(F1);          END;
PROCESS E: BEGIN;                      Taskbody(E); V(F2);          END;
PROCESS F: BEGIN; P(F1); P(F2);        Taskbody(F); V(I3);          END;
PROCESS G: BEGIN; P(G);                Taskbody(G); V(I2);          END;
PROCESS H: BEGIN; P(H);                Taskbody(H); V(I1);          END;
PROCESS I: BEGIN; P(I1); P(I2); P(I3); Taskbody(I); V(K);           END;
PROCESS K: BEGIN; P(K);                Taskbody(K);                 END;
PROCESS M: BEGIN;                      Taskbody(M); V(D2);          END;
```

Synchronisieren Sie jetzt mit Hilfe von Semaphoren die Prozesse gemäß dem Präzedenzgraphen. Nehmen Sie die notwendigen Änderungen direkt im angegebenen Pseudocode vor. (6 Punkte)

Name:	Matrikelnummer:

Aufgabe 6 (Sicherheit, Cache) (15 Punkte)

a) *Gegeben sind 20 Benutzer und 100 Dateien/Objekte. Wie viele Einträge hat eine ACL und wie viele eine Rollenliste maximal?* *(4 Punkte)*

Eine ACL hat für jeden Benutzer max.einen Eintrag, also max 20.
Eine Rollenliste hat max. einen Eintrag für jede Datei, also max 100.

b) *Nennen Sie die vier vom POSIX-6-Komitee vorgeschlagenen Maßnahmen zur Sicherheit von Dateien/Objekten.* *(4 Punkte)*

1. least privilege,
2. audit trail,
3. ACL's und
4. mandatory access control.

c) *Nennen Sie die drei möglichen Maßnahmen um Cache-Inkonsistenz bei Speicherreferenzen zu vermeiden und vergleichen Sie diese mit den vier möglichen Maßnahmen zur Cache-Kohärenz bei Dateisystemen in Netzwerken.* *(7Punkte)*

Bei Speicherreferenzen:

- write back: Der Cache schreibt selbst die Daten zurück.
- write through: Der Prozessor schreibt ohne den Cache die Daten in den Hauptspeicher.
- Snooper (copy back): Der *snooper* überwacht ob im Cache befindliche Speicheradressen verändert werden und markiert diese gegebenenfalls.

Bei Netzwerken:

- write through: Nicht nur die Kopie der Datei im Cache wird geändert, sondern auch gleich die Datei auf dem Server.
- delayed write: Veränderungen werden in einem Paket vorgenommen und nicht einzeln.
- zentrale Kontrolle: Der Cache prüft ob das original verändert wurde und aktualisiert gegebenenfalls die Datei direkt vom Server.
- write on close: Die Änderungen an der original Datei werden erst nach dem Aufruf der close() Operation durchgeführt, somit hat man das Problem an die Sitzungssemantik verschoben.

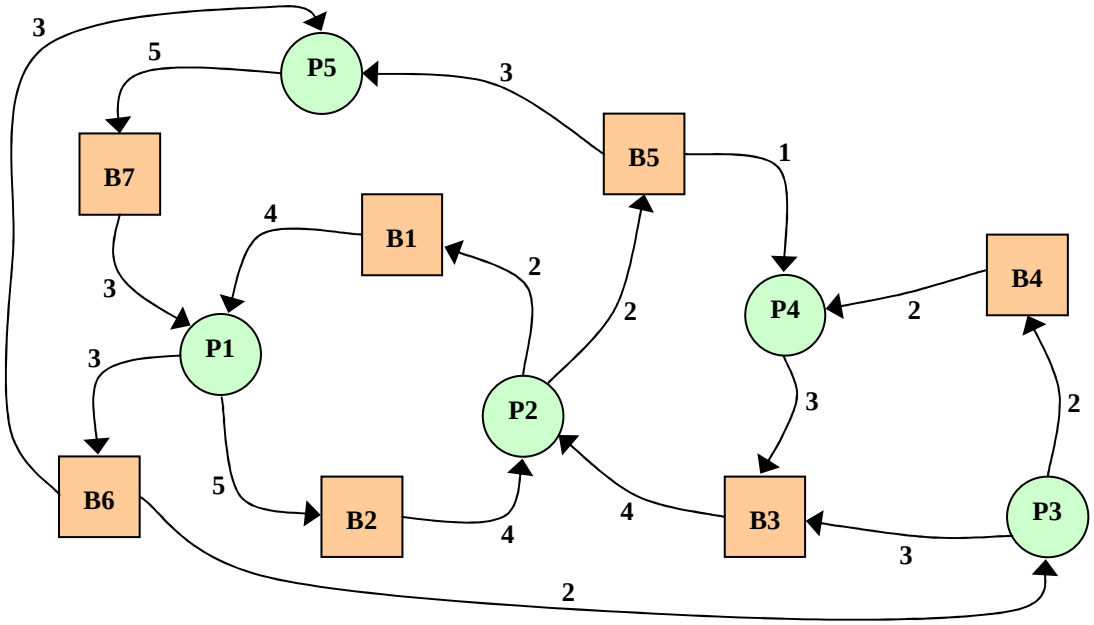
Abschließend kann man sagen, dass bei beiden dieselben Probleme bewältigt werden müssen. Dies merkt man daran, dass *snooper* im Endeffekt der zentralen Kontrolle entsprechen und *write through* bei beiden identisch ist. Allerdings gibt es auch Unterschiede, die durch die high-level Kontrolle und das Netzwerk bedingt sind, wie etwa *delayed write* und *write on closed*.

Name:	Matrikelnummer:

Aufgabe 7 (Verklemmungen) (15 Punkte)

*Gegeben sind die Prozesse **P1**, **P2**, **P3**, **P4**, **P5** und außerdem die folgende Betriebsmittel (die Zahlen in den Klammern geben an, wie viele davon jeweils insgesamt vorhanden sind): **B1** (4), **B2** (7), **B3** (6), **B4** (2), **B5** (6), **B6** (8), **B7** (9).*

*Weiter sei der folgende **Betriebsmittelgraph** gegeben:*



a) *Tragen Sie die durch den Betriebsmittelgraphen definierten Abhängigkeiten in die folgenden Tabellen ein. In die **Tabelle B** werden die von den Prozessen schon **belegten** Betriebsmittel eingetragen, in die **Tabelle C** die von den Prozessen noch **geforderten** Betriebsmittel:*

B	B1	B2	B3	B4	B5	B6	B7
P1	4						3
P2		4	4				
P3						2	
P4				2	1		
P5					3	3	

C	B1	B2	B3	B4	B5	B6	B7
P1		5				3	
P2	2				2		
P3			3	2			
P4			3				
P5							5

(3 Punkte)

b) *Versuchen Sie jetzt mögliche Verklemmungen zu erkennen. Führen dazu den Banker-Algorithmus durch. Notieren Sie in **jedem** Durchlauf wie das **Array A** aussieht **bevor** Sie einen Prozess markieren, dann welchen Prozess Sie markieren und warum, und zum Schluss noch einmal wie das **Array A nach** dem Durchlauf aussieht. Treten Verklemmungen auf? Falls ja, nennen sind die verklemmten Prozesse.* *(4 Punkte)*

- alle Prozess unmarkiert, A = {0,3,2,0,2,3,6}

- Durchlauf 1:
Wähle Prozess 5, da nur er laufen kann. Er gibt alle seine Betriebsmittel frei, das ergibt:
A = {0,3,2,0,5,6,6}

- Durchlauf 2:
Kein Prozess kann mehr Betriebsmittel freigeben, d.h. alle restlichen Prozesse sind verklemmt.

c) *Nennen Sie die vier notwendigen Bedingungen für das Auftreten von Verklemmungen.* *(4 Punkte)*

mutual exclusion
hold and wait,
circular wait,
no preemption,

d) *Nennen Sie die vier in der Vorlesung vorgestellten Vorgehensweisen, mit denen man das Problem der Verklemmungen behandeln könnte.* *(4 Punkte)*

Verklemmungen ignorieren
Verklemmungen vermeiden/verhindern
Verklemmungen unmöglich machen
Verklemmungen erkennen/beseitigen

Name:	Matrikelnummer:

Aufgabe 8 (Speicher) (20 Punkte)

Gehen Sie von 12 Bit virtuellen Speicheradressen aus. Die zwei höchstwertigsten Bits kodieren dabei den Index in der Basisseitentabelle. Die zwei nachfolgenden Bits kodieren den Index in der entsprechenden Tafel. Die letzten 8 Bits kodieren den *Offset*. Die physikalischen Adressen sind in hexadezimaler Notation angegeben.

3	1
2	2
1	3
0	0

Basistabelle

3	4
2	9
1	D
0	3

Tafel 0

3	7
2	E
1	1
0	C

Tafel 1

3	F
2	2
1	B
0	6

Tafel 2

3	0
2	5
1	8
0	A

Tafel 3

a) *Wie groß ist eine (physikalische) Speicherseite?* *(2 Punkte)*

Da der *Offset* 8 Bit beträgt und wir von Speichereinheiten der Größe 1 Byte ausgehen, haben wir 256Byte für eine Speicherseite.

Wie viele physikalische Speicherseiten können mit dieser vorgestellten virtuellen Adressierung adressiert werden? *(2 Punkte)*

Mit 2 Bit können wir genau 4 Tafeln adressieren und mit den nächsten 2 Bit die 4 Einträge innerhalb dieser Tafeln, also insgesamt 4*4 = 16 Seitenadressen.

b) Berechnen Sie für die folgenden sechs virtuellen Speicheradressen (in hexadezimaler Notation) die entsprechenden physikalischen Speicheradressen (in hexadezimaler Notation): *(6 Punkte)*

- 4EF = **0100**EF => BasistafelEintrag 1 = Tafel 3, Eintrag 0 = A => phys. Adresse = **AEF**
- 3A9 = **0011**A9 => BasistafelEintrag 0 = Tafel 0, Eintrag 3 = 4 => phys. Adresse = **4A9**
- B78 = **1011**78 => BasistafelEintrag 2 = Tafel 2, Eintrag 3 = F => phys. Adresse = **F78**
- C32 = **1100**32 => BasistafelEintrag 3 = Tafel 1, Eintrag 0 = C => phys. Adresse = **C32**
- DB7 = **1101**B7 => BasistafelEintrag 3 = Tafel 1, Eintrag 1 = 1 => phys. Adresse = **1B7**
- 5CD = **0101**CD => BasistafelEintrag 1 = Tafel 3, Eintrag 1 = 8 => phys. Adresse = **8CD**

Name:	Matrikelnummer:

c) *Sei* ***S*** *die Größe einer Speicherseite und seien die Speicheranforderungen im Intervall von 0 bis* $\frac{S}{2}$ *gleichverteilt. Wie groß ist der erwartete Verschnitt?* *(5 Punkte)*

Die erwartete Größe einer Speicheranforderung ist $\frac{S}{4}$, da die Anforderungen gleichverteilt sind.

Daraus berechnet sich der Verschnitt wie folgt: Verschnitt = S-$\frac{S}{4}$ = $\frac{3}{4}S$.

d) *Nennen Sie fünf Ihnen bekannte Speicherbelegungsstrategien.* *(5 Punkte)*

First-Fit,
Next-Fit,
Best-Fit,
Worst-Fit,
Quick-Fit,
Buddy-Systeme.

Name:	Matrikelnummer: