1. Aufgabe 1: Prozessscheduling

Teil a

FCFS:
Warteschlange: 1, 2, 3, 4, 5
mittlere Wartezeit: $4 * 5 + 3 * 7 + 2 * 3 + 1 * 8 = 55 \Rightarrow 11$

LCFS:
Warteschlange: 5, 4, 3, 2, 1
mittlere Wartezeit: $4 * 2 + 3 * 8 + 2 * 3 + 1 * 7 = 45 \Rightarrow 9$

Teil b

RR (ZQ=5):

CPU	**Prozess**	**Wartend**
0- 5	1	2,3,4,5
5-10	2	3,4,5
10-13	3	4,5,2
13-18	4	5,2
18-20	5	2,4
20-22	2	4
22-25	4	--

mittlere Wartezeit: $4 * 5 + 3 * 5 + 3 * 3 + 2 * 5 + 2 * 2 + 1 * 2 = 60 \Rightarrow 12$

2. Aufgabe 2: Prozesssynchronisation

Das Programm enthält in Zeile 03 eine Endlosschleife.
Daher gilt:

- Mutual Exclusion ist erfüllt.
- Progress Requirement und Bounded Waiting sind nicht erfüllt.

3. Aufgabe 3: Prozesszustände

```
    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
P1 ne ru ru ru ru ru wa wa wa wa re ru te  -  -  -  -  -  -  -  -  -
P2  -  - ne re re re re ru ru ru ru bl bl bl wa wa wa wa ru ru ru te
P3  - ne re re re re re re re re re re re ru ru ki  -  -  -  -  -  -  -
P4  -  -  -  - ne re ru bl bl bl wa wa wa wa ru ru te  -  -  -  -  -
```

4. Aufgabe 4: Wechselseitiger Ausschluss

Teil a

```
init (S,1);
hoch() {
      wait(S);

      if(A < 20) ++A;
      else if(B < 20) ++B;
      else if(C < 40) ++C;

      signal(S);
}
```

Teil b

```
runterA() {
      wait(S);
      if(A > 0) --A;
      signal(S);
}

runterB() {
      wait(S);
      if(B > 0) --B;
      signal(S);
}

runterC() {
      wait(S);
      if(C > 0) --C;
      signal(S);
}
```

5. Aufgabe 5: Speicherverwaltung

Teil a: 320+58+120+60+150+110=818

Teil b:
kleinste Adresse: 310
größte Adresse: 1639

Teil c:
780 (3,30)
1436 (0,116)
580 (1,18)
984 segfault
420 (2,110)

6. Aufgabe 6: Speicherverwaltung

m = 3

String	5	3	1	2	3	1	3	3	1	4	5	2	5	4	3	2	1	1	2	3	4
PF				*						*	*	*			*	*	*				*
$'''_1$	5	3	1	2	3	1	3	3	1	4	5	2	5	4	3	2	1	1	2	3	4
$'''_2$		5	3	1	2	3	1	1	3	1	4	5	2	5	4	3	2	2	1	2	3
m_3			5	3	1	2	2	2	2	3	1	4	4	2	5	4	3	3	3	1	2

m = 4:

String	5	3	1	2	3	1	3	3	1	4	5	2	5	4	3	2	1	1	2	3	4
PF										*	*	*			*		*				
m_1	5	3	1	2	3	1	3	3	1	4	5	2	5	4	3	2	1	1	2	3	4
m_2		5	3	1	2	3	1	1	3	1	4	5	2	5	4	3	2	2	1	2	3
m_3			5	3	1	2	2	2	2	3	1	4	4	2	5	4	3	3	3	1	2
m_4				5	5	5	5	5	5	2	3	1	1	1	2	5	4	4	4	4	1

7. Aufgabe 7: Wechselseitiger Ausschluss

Das Verfahren arbeitet nicht korrekt:

a) Mutual Exclusion wird nicht eingehalten:

Angabe eines möglichen Schedules, der eine Möglichkeit für beide Prozesse zeigt, gleichzeitig in den kritischen Bereich zu kommen.

```
1  // initial: turn=i, flag[i]=flag[j]=false;
2  // P_i                      P_j
3
4                             flag [j ]=true;
5                             while ( turn != j) {
6                               while ( flag [j]);
7    flag [i]=true;
8    while ( turn != i)
9    criticalSection (i);
10                            turn =j;
11                            }
12                            criticalSection (j)
13                            ...
14   ...
```

Dann sind beide Prozesse in ihrem kritischen Bereich.

b) Bounded Waiting wird nicht eingehalten:

```
1  // initial: turn=i, flag[i]=flag[j]=false;
2  // P_i                      P_j
3
4                             flag [j ]=true;
5                             while ( turn != j)
6                             criticalSection (j)
7    flag [i]=true;
8    while ( turn != i)  {
9      while ( flag [j]);
10                            flag [j ]=false;
11                            remainderSection (j)
12                            flag [j ]=true;
13     // Pi bekommt CPU
14     // und scheitert
15     // nach wie vor
16     // an while-Bed.
17                            while ( turn != j)
18                            criticalSection (j)
19                            ...
```

Pi kann im obigen Szenario beliebig oft überholt werden.

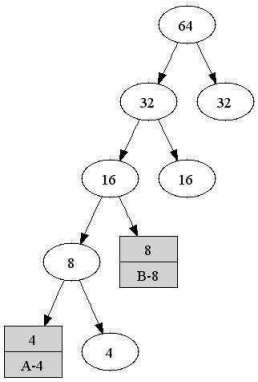
c) Progress Requirement wird eingehalten:

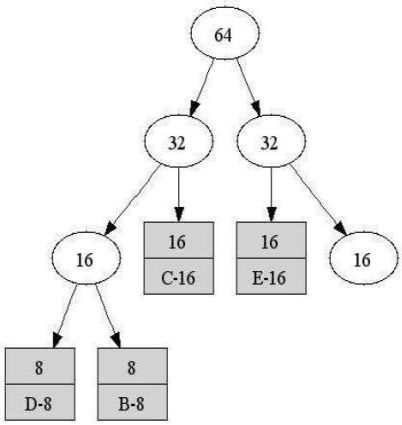
Sobald ein Prozess sein Flag auf *true* gesetzt hat, ist es für den anderen nicht mehr möglich, die turn Variable zu andern. Gilt oBdA initial turn == i, und Pi betritt als erster die Schleife, so gelangt Pi immer in seinen KB, da Pj turn nicht mehr ändern kann. Gilt initial turn = j, und Pi betritt als erster die Schleife, so hat er sein Flag auf true gesetzt. Entweder er kann die turn Variable auf i setzen, und gelangt danach in den KB, oder er wird von Pj überholt (siehe Teil b)), und Pj gelangt in den KB. Eine Blockade ist aber nicht möglich.

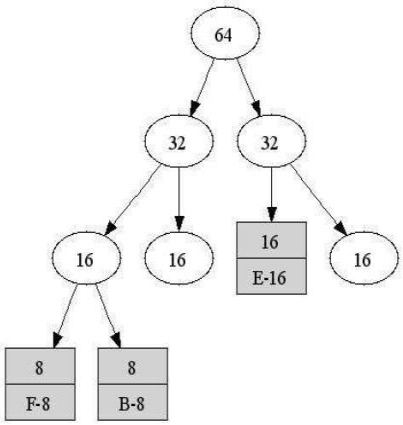
8. Aufgabe 8: (Buddy-Systeme)

In den Lösungen sind die wichtigsten Veränderungen in den Buddysystemen eingetragen.

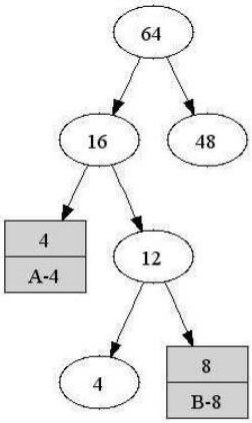
Teil **a**)

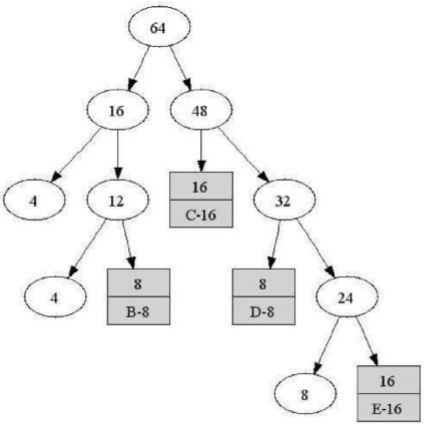


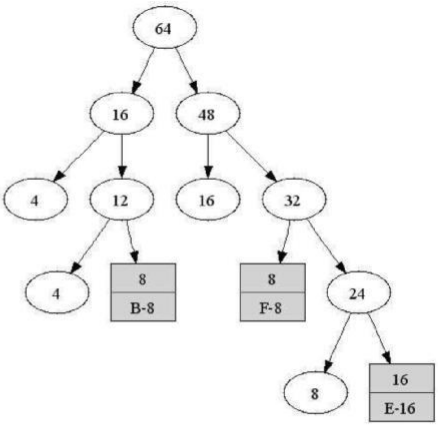




Teil **b**)

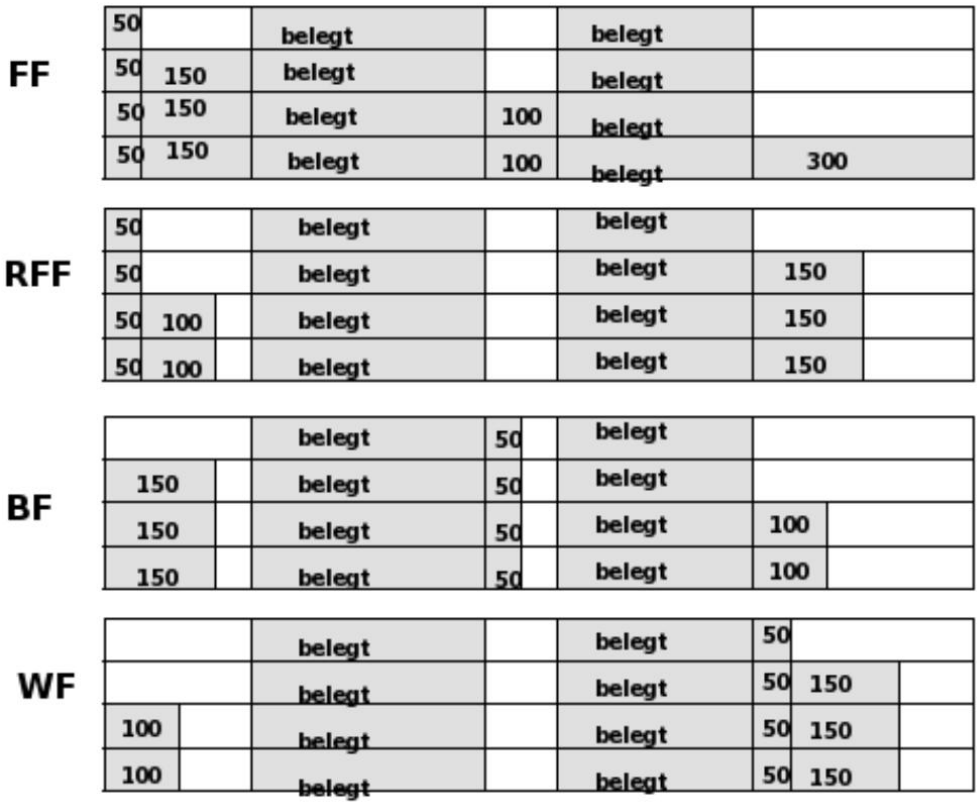






9. Aufgabe 9: (Segmentierung)

Teil a)



Teil b)

Zunächst einige Vorüberlegungen. Beispielsweise wird BF immer schiefgehen, sobald die 50 zuerst angefordert wird. WF geht immer schief, wenn die 300 nicht als erstes angefordert wird. Man erhält als eine Lösung.

- FF: Wir haben oben gesehen, dass die ursprünglichen Reihenfolgen nur für FF funktionieren.

- BF: originale Speicherreihenfolge, erste Anforderung 100, Rest beliebig.

- WF: Anforderungen in der Abfolge 300, 50, 150, 100; Speicher beginnt mit 100, danach beliebig.

- RFF: Anforderungen in der Abfolge 50, 100, 300, 150; Speicher wie im Original.

10.Aufgabe 10: (Segmentierung)

Lösung analog vorherige Aufgabe

11.Aufgabe 11: (Paging)

Teil **a**)

First In First Out (FIFO)

ω	2	5	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9	3	5	1
	2	5	3	4	4	6	8	9	7	0	0	0	4	3	5	8	9	9	9	1
		2	5	3	3	4	6	8	9	7	7	7	0	4	3	5	8	8	8	9
			2	5	5	3	4	6	8	9	9	9	7	0	4	3	5	5	5	8
				2	2	5	3	4	6	8	8	8	9	7	0	4	3	3	3	5
						2	5	3	4	6	6	6	8	9	7	0	4	4	4	3
							X	X	X	X			X	X	X	X	X			X

10PF

Teil **b**)
LRU

ω	2	5	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9	3	5	1
	2	5	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9	3	5	1
		2	5	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9	3	5
			2	5	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9	3
				2	2	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9
						2	3	4	5	6	6	6	7	0	8	9	4	4	4	8
							X	X	X	X			X	X	X					X

8PF

Teil **c**)

Als Auswahlkriterium wird das Alter seit dem Beginn des Referenzstrings angewandt. In der Lösung wird die Anzahl der Zugriffe auf die Seiten in der jeweiligen zweiten Spalte angezeigt.

LFU

ω	2	5	3	4	5	6	8	9	7	0	8	9	4	3	5	8	9	3	5	1
	2|0	5|0	3|0	4|0	4|0	6|0	8|0	9|0	7|0	0|0	0|0	0|0	4|0	3|0	3|0	3|0	3|0	3|1	3|1	1|0
		2|0	5|0	3|0	3|0	4|0	6|0	8|0	9|0	7|0	7|0	7|0	0|0	4|0	4|0	4|0	4|0	4|0	4|0	3|1
			2|0	5|0	5|1	3|0	4|0	6|0	8|0	9|0	9|0	9|1	9|1	9|1	9|1	9|1	9|2	9|2	9|2	9|2
				2|0	2|0	5|1	3|0	4|0	6|0	8|0	8|1	8|1	8|1	8|1	8|1	8|2	8|2	8|2	8|2	8|2
						2|0	5|1	5|1	5|1	5|1	5|1	5|1	5|1	5|1	5|2	5|2	5|2	5|2	5|3	5|3
							X	X	X	X			X	X						X

7PF

12. Aufgabe 12: (Scheduling)

Teil **a)**

Strat.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	WZ
FIFO	P_1		P_2					P_3							P_4								P_5			P_6		P_7							13.9
LIFO	P_7							P_6	P_5			P_4							P_3									P_2					P_1		14.4
SJF	P_1		P_6		P_5			P_2					P_7							P_3							P_4							9.7
RR	P_1		P_2					P_3					P_4					P_5		P_6			P_7					P_3		P_4		P_7		16

Teil **b)**

Prozess	CPU-Einheiten	Wartezeit
P_1	1-2	0
P_2	3-7	2
P_3	8-12, 28-29	7+15
P_4	13-17, 30-32	12+12
P_5	18-20	17
P_6	21-22	20
P_7	23-27, 33	22+5