1. Aufgabe (Prozessscheduling)

Wir betrachten im folgenden fünf Prozesse, die alle auf einer CPU bearbeitet werden müssen. Die Länge der Prozesse ist in Klammern angegeben: *P1*(5), P2(7), P3(3), P4(8), P5(2). Alle Prozesse sind bereits zum Zeitpunkt *t = 0* im System vorhanden und sind nach aufsteigendem Index angekommen (also *P1* zuerst).

a) Geben Sie die entstehenden Warteschlangen für die Strategien FCFS und LCFS an und bestimmen Sie jeweils die mittlere Wartezeit

b) Betrachten Sie nun das Round-Robin Verfahren (Zeitquantum 5). Geben Sie zu jedem Prozess an, zu welchen Zeitpunkten er die CPU belegt! Die Umschaltzeiten sollen hierbei vernachlässigt werden. Bestimmen Sie auch hier die mittlere Wartezeit!

2. Aufgabe (Prozesssynchronisation)

Gegeben seien zwei Prozesse P0 und P1. Es existiert eine kritische Sektion, die nicht von beiden Prozessen gleichzeitig betreten werden kann. Folgender Synchronisationsalgorithmus wird vorgeschlagen:

Den Prozessen P0 und P1 werden folgende gemeinsame Variable zur Verfügung gestellt:

```
bool flag[1];         /* mit FALSE initialisiert */
```

Der Prozess Pi arbeitet nach folgender Programmsequenz, wobei Pj den jeweils konkurrierenden Prozess bezeichnet:

```
01 repeat
02   flag[j]=TRUE;
03   while (flag[j]) do noop;
04   critical_section(Pi);
05   flag[i]:=FALSE;
06   remainder_section(Pi);
07 until FALSE;
```

Arbeitet das Verfahren korrekt? Prüfen Sie dies anhand der Anforderungen an eine korrekte Lösung des wechselseitigen Ausschlussproblems. Skizzieren Sie für jede der Bedingungen den Beweis oder geben Sie ein Gegenbeispiel an!

3. Aufgabe (Prozesszustände)

In der Vorlesung haben Sie gelernt, dass sich Prozesse während ihrer Lebensdauer in verschiedenen Zuständen befinden können:

Zustand	**Bedeutung**
new	Prozess wird erzeugt (kreiert)
running	Anweisungen des Prozesses werden gerade ausgeführt
ready	Prozess ist bereit zur Ausführung, wartet aber noch auf freien Prozessor
waiting	Prozess wartet auf das Eintreten eines Ereignisses, z.B. darauf, dass ein von ihm belegtes Betriebsmittel fertig wird
blocked	Prozess wartet auf ein fremdbelegtes Betriebsmittel
killed	Prozess wird (vorzeitig) abgebrochen
terminated	Prozess ist beendet, d.h. alle Instruktionen sind abgearbeitet

Gegeben seien 4 Prozesse, die zu unterschiedlichen Zeitpunkten gestartet werden und sich in die ready-Warteschlange einreihen. Der Scheduler arbeitet nach dem nicht-preemtiven LIFO-Verfahren.
Einige Prozesse benötigen Zugriff auf den Drucker. Die zweite Zeile der nachfolgenden Tabelle gibt an, nach wie vielen Zeiteinheiten im Zustand running ein Prozess auf den Drucker zugreifen möchte. Während der 4 Zeiteinheiten dauernden Druckphase wird der Prozessor für andere Prozesse freigegeben. Nach Beendigung des Druckvorgangs reiht sich der Prozess wieder in die ready-Warteschlange ein. Der Prozess P3 wird abgebrochen,
nachdem er 2 Zeiteinheiten gerechnet hat.

Prozess	P1	P2	P3	P4
Startzeitpunkt	0	2	1	4
Drucken nach Zeiteinheit	5	4	-	1
Benötigte Zeiteinheiten im Zustand *running*	6	7	10	3

Geben sie zu den ersten 22 Zeiteinheiten die aktuellen Prozesszustände der Prozesse P1 bis P4 an. Beachten Sie, dass ein ankommender Prozess bei seiner Ankunft den Zustand *new* annimmt und erst nach einer Zeiteinheit in einen anderen Zustand wechselt, wobei keine CPU-Zeit verbraucht wird. Des Weiteren können Prozesse, die ihren Druckvorgang beendet haben, direkt in den Zustand *running* wechseln, ohne vorher eine Zeiteinheit im Zustand *ready* verbringen zu müssen.

4. Aufgabe (Wechselseitiger Ausschluss)

Der Eiffelturm in Paris besitze 3 Aussichtsplattformen: die nebeneinander liegenden Plattformen A und B sowie die etwas tiefer gelegene Plattform C. Die beiden Plattformen A und B können jeweils 20 Personen aufnehmen. Die tiefer gelegene Plattform C hat eine Kapazität von 40 Personen. Da die höher gelegenen Plattformen touristisch attraktiver sind, werden zunächst alle Touristen auf die Plattformen A und B transportiert. Sind diese voll, werden die restlichen Kunden zur Plattform C gebracht. Die Touristen werden mit mehreren Aufzügen nach oben bzw. wieder nach unten gebracht. Jeder Aufzug kann nur eine Person befördern.

a) Schreiben Sie einen Algorithmus hoch() in Pseudocode, der die Touristen auf die Plattformen befördert. Berücksichtigen Sie dabei die oben genannten Kapazitäten und die oben beschriebene Strategie. Verwenden Sie Semaphoren mit assoziierter Warteschlange.

b) Schreiben Sie drei Algorithmen runterA(), runterB(), runterC(), welche die Touristen von den Plattformen wieder zurückbringen.

5. Aufgabe (Speicherverwaltung)

Für die Speicherverwaltung nach dem Segmentierungsverfahren sei für ein Programm die folgende Segmenttabelle gegeben (Länge in Speicherworten):

Segment	Basis	Länge
0	1320	320
1	562	58
2	310	120
3	750	60
4	830	150
5	990	110

a) Wieviele Speicherworte stehen dem Programm im physikalischen Speicher zur Verfugung?¨

b) Welches ist die kleinste und welches die größte für das Programm verfügbare physikalische Adresse?

c) Berechnen Sie zu den folgenden physikalischen Adressen jeweils die logischen Adressen. Welche Anfrage löst einen Segmentation Fault aus?

1. 780
2. 1436
3. 580
4. 984
5. 420

6. Aufgabe (Speicherverwaltung)

Der Referenz String ω beschreibe die Folge der Seitenzugriffe. Die Tabellen zeigen die Belegung der Seitenrahmen im Hauptspeicher. Es stehen einmal drei und einmal vier Rahmen zur Verfügung. Vervollständigen Sie die Tabellen unter Verwendung der LRU Strategie. Markieren Sie die Seitenfehler mit einem ⋆.

ω =	5	3	1	2	3	1	3	3	1	4	5	2	5	4	3	2	1	1	2	3	4

Seitenfehler				*																	
Seitenrahmen 1	5	3	1	2																	
Seitenrahmen 2		5	3	1																	
Seitenrahmen 3			5	3																	

Seitenfehler																					
Seitenrahmen 1	5	3	1	2																	
Seitenrahmen 2		5	3	1																	
Seitenrahmen 3			5	3																	
Seitenrahmen 4				5																	

7. Aufgabe (Wechselseitiger Ausschluss)

Eine Lösung des Problems des wechselseitigen Ausschlusses von zwei Prozessen wurde 1966 von L. Hyman vorgestellt. Der Algorithmus stellt den Prozessen P_0 und P_1 folgende gemeinsame Variablen zur Verfügung

```
bool flag [2]; // mit false initialisiert
int turn ;   // mit den Werten 0 bis 1, Initialisierung beliebig
```

Der Prozess Pi arbeitet nach folgender Programmsequenz, wobei Pj den jeweils konkurrierenden Prozess bezeichnet:

```
 1 while (true)
 2  {
 3   flag[i] = true;
 4   while (turn != i)
 5     {
 6        While (flag[j])
 7        {
 8           noop;
 9        }
10        turn=i;
11     }
12     criticalSection(Pi);
13     flag[i] = false;
14     remainderSection(Pi);
15 }
```

Arbeitet das Verfahren korrekt? Prüfen Sie dies anhand der Anforderungen an eine korrekte Lösung des wechselseitigen Ausschlussproblems. Skizzieren Sie für jede der Bedingungen den Beweis oder geben Sie ein Gegenbeispiel an.

8. Aufgabe (Buddy-Systeme)

Der Hauptspeicher eines Systems sei 64 MB groß und werde mit dem Buddy-System verwaltet. Der Reihe nach werden folgende Speicherbereiche angefordert bzw. freigegeben.

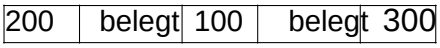
Operation	Name	Größe
Anforderung	A	4 MB
Anforderung	B	8 MB
Freigabe	A	
Anforderung	C	16 MB
Anforderung	D	8 MB
Anforderung	E	16 MB
Freigabe	D	
Anforderung	F	8 MB
Freigabe	C	

a) Stellen Sie die einzelnen Schritte in einem Baum dar (bei freier Wahl wird linker Ast gewählt).

b) Stellen sie die einzelnen Schritte nun für gewichtete Buddies dar (bei durch 3 teilbarer Speichergröße Gewichtung 1:2, sonst Gewichtung 1:3).

9. Aufgabe (Segmentierung)

In dieser Aufgabe geht es um die Speicherverwaltung mit Segmentierung. Gegeben sei folgende Belegung eines Speichers mit Lücken der Größe 200, 100 und 300 kB:



Weiterhin seien vier Belegungensmethoden für Speicherplatzanforderungen gegeben: First-Fit, Rotating-First-Fit, Best-Fit, Worst-Fit.

a) Wie sieht der obige Speicher aus, wenn nacheinander vier Anforderungen der Größe 50, 150,100 und 300 kB ankommen? Notieren Sie für jede Strategie die freien Speicherbereiche nach jeder Anforderung (z.B. (200, 100, 300) für die obige Belegung), und geben Sie an, für welche Methoden die Anforderungen erfüllt werden können. Beachten Sie, dass die Daten jeweils linksbündig in einer Lücke abgelegt und einmal belegte Speicherbereiche nicht wieder freigegeben werden.

b) Sie dürfen nun die Reihenfolge der freien Speicherbereiche und die der Anforderungen vertauschen (aber keine Speicherbereiche zusammenfassen). Geben Sie für jede Strategie eine Speicherbelegung und die zugehörigen Anforderungen an, die für jeweils nur diese Methode erfolgreich arbeitet und für die übrigen Methoden nicht alle Anforderungen erfüllen kann. Geben Sie auch die Speicherbelegungen nach Abar-beitung der einzelnen Anforderungen an.

10. Aufgabe (Segmentierung)

Gegeben sei eine Liste freier Speicherbereiche: 100kB - 400kB - 250kB - 200kB - 50kB. Wie verhalten sich jeweils die Strategien **first fit, next fit, best fit** und **worst fit** auf die nacheinander eingehenden Speicheranforderungen 30kB, 60kB, 120kB, 20kB, 100kB, 250kB?

Halten Sie dabei den Zustand nach jeder Speicheranforderung fest.

11. Aufgabe (Paging)

In dieser Aufgabe geht es um die Speicherverwaltung mit Paging. Der Hauptspeicher eines Systems habe 5 Frames, die alle zu Beginn leer seien. Gegeben sei der Referenzstring

$$\omega = 25345689708943589351.$$

Geben Sie für jede der folgenden Strategien die Belegung der Speicherstellen zu jedem Zeitpunkt und die Anzahl der Seitenfehler an. Orientieren Sie sich an die Definitionen und Vorgaben aus den Folien der Vorlesungen.

a) FIFO

b) LRU

c) LFU

12.Aufgabe (CPU-Scheduling)

Gegeben seien die Prozesse $P_1, \ldots, P_7$, die alle auf einer CPU bearbeitet werden müssen. Die Länge der Prozesse ist der folgenden Tabelle zu entnehmen:

Prozess	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Länge	2	5	7	8	3	2	6

Zum Zeitpunkt t = 0 sind alle Prozesse bereits im System vorhanden und sind nach aufsteigendem Index angekommen (also P_1 zuerst).

a) Geben Sie die entstehenden Warteschlagen für die Strategien FIFO, LIFO und SJF (*Shortest Job First*) an und bestimmen Sie zu jeder Strategie die mittlere Wartezeit.

b) Wenden Sie nun das Round-Robin Verfahren für die obigen Prozesse und mit einem Zeitquantum von 5 an. Geben Sie für jeden Prozess an, zu welchen Zeitpunkten er die CPU belegt. Dabei sollen die Umschaltzeiten vernachlässigt werden. Bestimmen Sie auch hier die mittlere Wartezeit.

13.Aufgabe (Virtulle Adressierung)

Die Breite einer virtuellen Adresse betrage 12 Bit. Als physikalischer Speicher stehen 256 Byte zur Verfügung.

a.) Zählen Sie alle möglichen Seitengrößen auf, bei denen der physikalische Speicher komplett genutzt wird!

b.) Nehmen Sie im weiteren eine Seitengröße von 32 Byte an. Wieviel Speicher (Hintergrund-speicher, physikalischer Speicher, evtl. Puffer) müssen für den Betrieb mindestens zur Verfügung stehen?

c.) Wie viele Bits der virtuellen Adresse entfallen auf die Seitennummer und wieviel auf den Offset?