Name, Vorname: _____

Matrikelnr.: _____

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Summe**
maximale Punktzahl	9	9	12	16	12	14	12	16	0
erreicht									

Organisatorische Hinweise

- Dauer der Klausur: 90 Minuten
- Erlaubte Hilfsmittel:
  - Ein DIN A4-Blatt mit handschriftlichen Notizen (Vorder- und Rückseite beschrieben)
  - Taschenrechner (ohne Mobilfunk- / ohne Internetanschluss, kein Smartphone)

Verhalten während der Klausur

- Beschriften Sie sofort das Deckblatt mit Ihrem Namen und Ihrer Matrikel-Nr.!
- Verwenden Sie die vorgegebenen Klausuraufgabenzettel für Ihre Lösungen (ggf. auch die Rückseite).
- Benutzen Sie keinen Bleistift und keine rote Tinte. Entsprechende Aufzeichnungen werden nicht gewertet.
- Legen Sie Ihren Studierendenausweis gut sichtbar auf den Tisch.
- Sofern Sie ein (max. DinA4-)Hilfsblatt vorgefertigt haben legen Sie es zum Ausweis.
- Melden Sie sich bitte mit Handzeichen, wenn Sie zur Toilette müssen (zeitgleich darf nur ein Klausurteilnehmer den Raum verlassen)
- Vermeiden Sie möglichst alle Störungen!

Aufgabe 1

a. Was ist ein Interrupt?

b. Welche zwei grundsätzlichen Arten von Interrupts gibt es?

1. _____

2. _____

c. Ein Interrupt welcher Art wird durch Ablauf eines Timers (Zeitscheibe) ausgelöst?

d. Ein Interrupt welcher Art wird durch Zugriff auf einen privilegierten Befehl im User-Modus ausgelöst? *(bei Prozessoren mit Virtualisierungsunterstützung)*

Welche Aufgabe hat in diesem Fall der Interrupt-Handler in einem virtuellen System mit Type-1-Hypervisor?

Aufgabe 2

a. Erläutern Sie kurz das „Cache“-Prinzip!

b. Nennen Sie eine Verhaltenweise von Programmen, die einen hohen Effizienzgewinn beim Einsatz eines Cache-Speichers ermöglicht!

c. Geben Sie zwei Beispiele für die Verwendung von Cache-Speicher in einem modernen Computer an! *(Stichworte genügen)*

1. _____

2. _____

Aufgabe 3

Gesucht ist der Code für eine spezielle **Unix**-Shell, die jeden eingegebenen gültigen Befehl **genau zweimal** ausführt, indem zweimal der gleiche Kindprozess gestartet wird. Der Shell-Prozess soll beide Kindprozesse starten und dann anschließend auf deren Beendigung warten.
Ergänzen Sie den angegeben (Pseudo-)C-Code!
Hinweise:

- *Auf das Abfangen von Fehlern dürfen Sie ausnahmsweise verzichten!*
- *Für den Aufruf eines Kindprozesses und für das Warten des Shell-Prozess auf Beendigung eines Kindes genügt ein Funktionsname ohne Parameter.*

```
while (TRUE) {
  type_prompt();
  read_command(..);

 /* Ihre Ergänzungen: */

}
```

Aufgabe 4

In dieser Aufgabe geht es um das Scheduling der CPU. Betrachtet wird ein Multilevel-Feedback-Queuing-Algorithmus, der 3 verschiedene Prioritätsklassen (nummeriert von 1-3) und die dynamische Neuberechnung von Prioritäten umfasst.

Es gilt Folgendes:

- Eine Zeitscheibe beträgt 30 ms.
- Ein Prozess darf seine begonnene Zeitscheibe immer voll ausnutzen.
- Eine höhere Prioritätsklasse hat auch eine höhere Dringlichkeit (3 hat also die höchste Dringlichkeit, d.h. die Prozesse in dieser Prioritätsklasse werden zuerst bearbeitet).
- Innerhalb einer Prioritätsklasse wird der Round-Robin-Algorithmus angewendet.
- Prozesswechselzeiten können vernachlässigt werden.
- Wenn ein Prozess 60 ms in seiner aktuellen Prioritätsklasse gewartet hat, erhöht sich seine Prioritätsklasse beim nächsten Scheduling-Zeitpunkt um 1 (bis maximal 3). Nach Ausführung einer Zeitscheibe wird seine Prioritätsklasse auf die angegebene Basis-Prioritätsklasse zurückgesetzt.

Gegeben sind folgende Prozesse mit Basis-Prioritätsklassen gemäß der folgenden Tabelle:

Prozessname	**Basis-Prioritätsklasse**
A	1
B	2
C	3

Alle drei Prozesse starten gleichzeitig bei 0 und laufen ohne Blockierung.

Geben Sie für den Zeitraum 0 – 240 ms an, welchen Prozess der Scheduler für die CPU-Zuteilung auswählt. Füllen Sie dazu die folgende Tabelle aus:

Scheduling-Zeitpunkt	**Aktuelle Prioritätsklasse A**	**Aktuelle Prioritätsklasse B**	**Aktuelle Prioritätsklasse C**	**Ausgewählter Prozess**
0 ms	1	2	3	
30 ms				
60 ms				
90 ms				
120 ms				
150 ms				
180 ms				
210 ms				
240 ms				

Aufgabe 5

Wir betrachten ein Java-Programm mit den folgenden zwei Klassen:

```
public class A5 {

  public static void main(String[] args) {
    LinkedList<MyThread> threadList = new LinkedList<>();
    for (int i = 0; i < 5; i++) {
       MyThread curThread = new MyThread();
       curThread.start();
       threadList.add(curThread);
     }
     try {
        Thread.sleep(50);



     } catch (InterruptedException e) { }
     System.err.println("-- Hauptthread wird beendet!--");
  }
}

class MyThread extends Thread {

   public void run() {
     System.err.println("Hallo, mein Name ist: " + getName());
     try {
        sleep(1000);
     } catch (InterruptedException e) {
     }
   }
}
```

a) Benötigt das angegebene Programm mehr als 2 Sekunden zur Ausführung (Laufzeitzeit, bis alle Threads beendet sind)? Begründen Sie Ihre Antwort!

b) Welche Auswirkungen hätte eine Veränderung der Zeile **`curThread.start();`**
in **`curThread.run();`** in der **`main`**-Methode der **`A5`**-Klasse bzgl. der gesamten Ausführungsdauer?

c) Ergänzen Sie oben entsprechenden Code, damit alle Threads der Klasse **`MyThread`** nach 50 ms unterbrochen werden!

Aufgabe 6

In der Hochschulbibliothek gibt es von Buch A zwei Exemplare, von Buch B ein Exemplar und von Buch C drei Exemplare. Vor der Bachelorarbeit muss ein Student von jedem dieser Bücher je ein Exemplar ausleihen (vorher kann er nicht mit der Bachelorarbeit beginnen), nach der Bachelorarbeit gibt er sie wieder zurück. Falls ein Buch bereits ausgeliehen ist, muss er warten, bis ein Exemplar zurückgegeben wird (Ausleihfristen werden hier nicht berücksichtigt).

a. Geben Sie einen Pseudocode-Abschnitt an, der den Prozess des Bücher-Ausleihens und Bachelorarbeit-Schreibens für einen Studenten mit Hilfe von Semaphoren beschreibt (inkl. Semaphor-Initialisierung)! Dieser Pseudocode muss dann von jedem Studenten ausgeführt werden.

b. Könnte es zu Synchronisationsproblemen kommen, wenn Studenten die Bücher in einer unterschiedlichen Reihenfolge (mehrere Code-Varianten) ausleihen?
Wenn ja: Geben Sie ein Beispiel für eine Fehlersituation an!

Aufgabe 7

Betrachten Sie einen seitenorientierten virtuellen Speicher mit einer Seitengröße von **4000 Bytes** („Paging“-Verfahren). Gegeben ist der folgende Ausschnitt der Seitentabelle eines Prozesses:

Virtuelle Seitennummer	0	1	2	3	4	5
Seitenrahmennummer	2	1	10	7	0	9
„Valid“-Flag	true	false	true	true	true	true
„Referenced“-Flag	true	false	true	true	false	true

a) Geben Sie für die im folgenden angegebenen virtuellen Adressen die virtuelle Seitennummer sowie die zugehörige reale Hauptspeicheradresse an (falls sich die Seite im Hauptspeicher befindet).

Virtuelle Adresse	456	5600	8000	9990

Virtuelle Seitennummer				
Reale Hauptspeicheradresse				

b) Welche der angegebenen virtuellen Adressen löst bei einem Zugriff einen Seitenfehler („Page fault“) aus?

c) Zur Ermittlung der nächsten zu ersetzenden Seite werde der Clock-Algorithmus verwendet, der eine eigene Liste aller momentan im Hauptspeicher befindlichen Seiten des Prozesses führt.
Die Liste sei hier (3,0,2,5,4), wobei die Nummer der virtuellen Seite, auf die der Uhrzeiger aktuell zeigt, am Anfang steht (also hier 3).
Betrachten Sie den Zugriff mit der virtuellen Adresse aus Teil b).
Welche virtuelle Seite wird bei Auslösen des Seitenfehlers aus dem Hauptspeicher verdrängt? Wie sieht anschließend die Liste für den Clock-Algorithmus aus?

Aufgabe 8

Wir betrachten ein kleines I-Node-basiertes Dateisystem (UNIX-ähnlich), das insgesamt 10 Blöcke für Benutzerdateien zur Verfügung stellt und das sich in folgendem Zustand befindet:

I-Node-Liste (Auszug):

I-Node-Nr.	Plattenblockverweise
0	7, 3, 0
1	3, 5
2	4, 6, 2

Freiliste (hier: Liste mit Blocknummern freier Blöcke):
(1, 3, 8)

Durch einen Systemabsturz ist das Dateisystem inkonsistent geworden. Sie wissen lediglich, dass alle I-Nodes korrekt durch mindestens einen Verzeichniseintrag referenziert werden.

a) Führen Sie einen Algorithmus aus, mit dem widersprüchliche Block-Informationen innerhalb der I-Node-Liste sowie zwischen I-Node-Liste und Freiliste ermittelt werden können.

b) Geben Sie alle gefundenen widersprüchlichen Informationen an und erklären Sie jeweils, wie Sie diesen Zustand bereinigen würden!

Name, Vorname: _____

Matrikelnr.: _____

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**Summe**
maximale Punktzahl	9	9	12	16	12	14	12	16	
erreicht									

Organisatorische Hinweise

- Dauer der Klausur: 90 Minuten
- Erlaubte Hilfsmittel:
  - Ein DIN A4-Blatt mit handschriftlichen Notizen (Vorder- und Rückseite beschrieben)
  - Taschenrechner (ohne Mobilfunk- / ohne Internetanschluss, kein Smartphone)

Verhalten während der Klausur

- Beschriften Sie sofort das Deckblatt mit Ihrem Namen und Ihrer Matrikel-Nr.!
- Verwenden Sie die vorgegebenen Klausuraufgabenzettel für Ihre Lösungen (ggf. auch die Rückseite).
- Benutzen Sie keinen Bleistift und keine rote Tinte. Entsprechende Aufzeichnungen werden nicht gewertet.
- Legen Sie Ihren Studierendenausweis gut sichtbar auf den Tisch.
- Sofern Sie ein (max. DinA4-)Hilfsblatt vorgefertigt haben legen Sie es zum Ausweis.
- Melden Sie sich bitte mit Handzeichen, wenn Sie zur Toilette müssen (zeitgleich darf nur ein Klausurteilnehmer den Raum verlassen)
- Vermeiden Sie möglichst alle Störungen!

Aufgabe 1

a. Was ist ein Interrupt?

Interrupt = Unterbrechung des Programmablaufs

b. Welche zwei grundsätzlichen Arten von Interrupts gibt es?

1. _____ asynchrone Interrupts_____

2. _____ synchrone Interrupts _____

c. Ein Interrupt welcher Art wird durch Ablauf eines Timers (Zeitscheibe) ausgelöst?

asynchroner Interrupt (durch externen Timer)

d. Ein Interrupt welcher Art wird durch Zugriff auf einen privilegierten Befehl im User-Modus ausgelöst? *(bei Prozessoren mit Virtualisierungsunterstützung)*

synchroner Interrupt (die CPU erzeugt eine Exception)

Welche Aufgabe hat in diesem Fall der Interrupt-Handler in einem virtuellen System mit Type-1-Hypervisor?

Aufruf des Hypervisors

Aufgabe 2

a. Erläutern Sie kurz das „Cache“-Prinzip!

Nach Zugriff auf ein langsameres Speichermedium werden die zuletzt gelesenen Daten im schnelleren Speicher gehalten („Cache-Speicher“). Vor erneutem Zugriff wird geprüft, ob Daten bereits im Cache vorhanden sind. Wenn ja, werden sie aus dem Cache gelesen statt vom langsameren Speichermedium.

b. Nennen Sie eine Verhaltenweise von Programmen, die einen hohen Effizienzgewinn beim Einsatz eines Cache-Speichers ermöglicht!

Lokalitätsverhalten (räumlich / zeitlich)

c. Geben Sie zwei Beispiele für die Verwendung von Cache-Speicher in einem modernen Computer an! *(Stichworte genügen)*

1. Prozessor-Cache (CPU-Cache) für Hauptspeicherzugriffe

2. Hauptspeicher-Cache für Festplattenzugriffe

Aufgabe 3

Gesucht ist der Code für eine spezielle **Unix**-Shell, die jeden eingegebenen gültigen Befehl **genau zweimal** ausführt, indem zweimal der gleiche Kindprozess gestartet wird. Der Shell-Prozess soll beide Kindprozesse starten und dann anschließend auf deren Beendigung warten.
Ergänzen Sie den angegeben (Pseudo-)C-Code!
Hinweise:

- *Auf das Abfangen von Fehlern dürfen Sie ausnahmsweise verzichten!*
- *Für den Aufruf eines Kindprozesses und für das Warten des Shell-Prozess auf Beendigung eines Kindes genügt ein Funktionsname ohne Parameter.*

```
while (TRUE) {
  type_prompt();
  read_command(..);

 /* Ihre Ergänzungen: */

      PID1 = fork();
      if (PID1 == 0) {
            /* 1. Kindprozess */
            exec (..);
      }
      PID2 = fork();
      if (PID2 == 0) {
            /* 2. Kindprozess */
            exec (..);
      }
      waitpid (PID1, ..);
      waitpid (PID2, ..);

}
```

Aufgabe 4

In dieser Aufgabe geht es um das Scheduling der CPU. Betrachtet wird ein Multilevel-Queuing-Feedback-Algorithmus, der 3 verschiedene Prioritätsklassen umfasst (nummeriert von 1-3) und eine dynamische Neuberechnung von Prioritäten umfasst.

Es gilt Folgendes:

- Eine Zeitscheibe beträgt 30 ms.
- Ein Prozess darf seine begonnene Zeitscheibe immer voll ausnutzen.
- Eine höhere Prioritätsklasse hat auch eine höhere Dringlichkeit (3 hat also die höchste Dringlichkeit, d.h. die Prozesse in dieser Prioritätsklasse werden zuerst bearbeitet).
- Innerhalb einer Prioritätsklasse wird der Round-Robin-Algorithmus angewendet.
- Prozesswechselzeiten können vernachlässigt werden.
- Wenn ein Prozess 60 ms in seiner aktuellen Prioritätsklasse gewartet hat, erhöht sich seine Prioritätsklasse beim nächsten Scheduling-Zeitpunkt um 1 (bis maximal 3). Nach Ausführung einer Zeitscheibe wird seine Prioritätsklasse auf die angegebene Basis-Prioritätsklasse zurückgesetzt.

Gegeben sind folgende Prozesse mit Basis-Prioritätsklassen gemäß der folgenden Tabelle:

Prozessname	Basis-Prioritätsklasse
A	1
B	2
C	3

Alle drei Prozesse starten gleichzeitig bei 0 und laufen ohne Blockierung.

Geben Sie für den Zeitraum 0 – 240 ms an, welchen Prozess der Scheduler für die CPU-Zuteilung auswählt. Füllen Sie dazu die folgende Tabelle aus:

Scheduling-Zeitpunkt	Aktuelle Priori-tätsklasse A	Aktuelle Priori-tätsklasse B	Aktuelle Priori-tätsklasse C	Ausgewählter Prozess
0 ms	1	2	3	C
30 ms	1	2	3	C
60 ms	2	3	3	B
90 ms	2	2	3	C
120 ms	3	2	3	A
150 ms	1	3	3	B
180 ms	1	2	3	C
210 ms	2	2	3	C
240 ms	2	3	3	B

Aufgabe 5

Wir betrachten ein Java-Programm mit den folgenden zwei Klassen:

```
public class A5 {

  public static void main(String[] args) {
    LinkedList<MyThread> threadList = new LinkedList<>();
    for (int i = 0; i < 5; i++) {
       MyThread curThread = new MyThread();
       curThread.start();
       threadList.add(curThread);
     }
     try {
        Thread.sleep(50);
        for (MyThread curThread : threadList) {
           curThread.interrupt();
        }
     } catch (InterruptedException e) { }
     System.err.println("-- Hauptthread wird beendet!--");
  }
```

```
}

class MyThread extends Thread {

   public void run() {
     System.err.println("Hallo, mein Name ist: " + getName());
     try {
        sleep(1000);
     } catch (InterruptedException e) {
     }
   }
}
```

a) Benötigt das angegebene Programm mehr als 2 Sekunden zur Ausführung (Laufzeitzeit, bis alle Threads beendet sind)? Begründen Sie Ihre Antwort!

Nein, da alle Threads parallel (bzw. nebenläufig) laufen und jeweils nur ca. 1 Sekunde Ausführungszeit benötigen!

b) Welche Auswirkungen hätte eine Veränderung der Zeile **`curThread.start();`** in **`curThread.run();`** in der **`main`**-Methode der **`A5`**-Klasse bzgl. der gesamten Ausführungsdauer?

Sequentielle Ausführung des gesamten Codes nur durch den Main-Thread, da keine BS-Threads erzeugt werden. Die Ausführungsdauer wird also mindestens 5 Sekunden betragen!

c) Ergänzen Sie oben entsprechenden Code, damit alle Threads der Klasse **`MyThread`** nach 50 ms unterbrochen werden!
s.o.

Aufgabe 6

In der Hochschulbibliothek gibt es von Buch A zwei Exemplare, von Buch B ein Exemplar und von Buch C drei Exemplare. Vor der Bachelorarbeit muss ein Student von jedem dieser Bücher je ein Exemplar ausleihen (vorher kann er nicht mit der Bachelorarbeit beginnen), nach der Bachelorarbeit gibt er sie wieder zurück. Falls ein Buch bereits ausgeliehen ist, muss er warten, bis ein Exemplar zurückgegeben wird (Ausleihfristen werden hier nicht berücksichtigt).

a. Geben Sie einen Pseudocode-Abschnitt an, der den Prozess des Bücher-Ausleihens und Bachelorarbeit-Schreibens für einen Studenten mit Hilfe von Semaphoren beschreibt (inkl. Semaphor-Initialisierung)! Dieser Pseudocode muss dann von jedem Studenten ausgeführt werden.

Pseudocode für einen Studenten-Prozess:

```
Allgemeines Semaphor A = 2;
Allgemeines Semaphor B = 1;
Allgemeines Semaphor C = 3;

P(A);
P(B);
P(C);
<Bachelorarbeit schreiben>
V(A);
V(B);
V(C);
```

b. Könnte es zu Synchronisationsproblemen kommen, wenn Studenten die Bücher in einer unterschiedlichen Reihenfolge (mehrere Code-Varianten) ausleihen?
Wenn ja: Geben Sie ein Beispiel für eine Fehlersituation an!

Ja! Beispiel:

Student 1 + 2:	Student 3:
P(A)	P(B)
P(B)	P(A)
...	...

Wenn Student 1 und 2 nach Ausleihen von Buch A unterbrochen werden und anschließend Student 3 an die Reihe kommt und Buch B ausleiht, warten anschließend alle jeweils auf das Buch, das der andere hat (Deadlock mit „Circular Wait"-Situation).

Aufgabe 7

Betrachten Sie einen seitenorientierten virtuellen Speicher mit einer Seitengröße von **4000 Bytes** („Paging"-Verfahren). Gegeben ist der folgende Ausschnitt der Seitentabelle eines Prozesses:

Virtuelle Seitennummer	0	1	2	3	4	5
Seitenrahmennummer	2	1	10	7	0	9
„Valid"-Flag	true	false	true	true	true	true
„Referenced"-Flag	true	false	true	true	false	true

a) Geben Sie für die im folgenden angegebenen virtuellen Adressen die virtuelle Seitennummer sowie die zugehörige reale Hauptspeicheradresse an (falls sich die Seite im Hauptspeicher befindet).

Virtuelle Adresse	456	5600	8000	9990
Virtuelle Seitennummer	0	1	2	2
Reale Hauptspeicheradresse	8456	–	40000	41990

b) Welche der angegebenen virtuellen Adressen löst bei einem Zugriff einen Seitenfehler („Page fault") aus? 5600

c) Zur Ermittlung der nächsten zu ersetzenden Seite werde der Clock-Algorithmus verwendet, der eine eigene Liste aller momentan im Hauptspeicher befindlichen Seiten des Prozesses führt.
Die Liste sei hier (3,0,2,5,4), wobei die Nummer der virtuellen Seite, auf die der Uhrzeiger aktuell zeigt, am Anfang steht (also hier 3).
Betrachten Sie den Zugriff mit der virtuellen Adresse aus Teil b).
Welche virtuelle Seite wird bei Auslösen des Seitenfehlers aus dem Hauptspeicher verdrängt? Wie sieht anschließend die Liste für den Clock-Algorithmus aus?

Seite 4 wird verdrängt → (3,0,2,5,1)

Aufgabe 8

Wir betrachten ein kleines I-Node-basiertes Dateisystem (UNIX-ähnlich), das insgesamt 10 Blöcke für Benutzerdateien zur Verfügung stellt und das sich in folgendem Zustand befindet:

I-Node-Liste (Auszug):

I-Node-Nr.	Plattenblockverweise
0	7, 3, 0
1	3, 5
2	4, 6, 2

Freiliste (hier: Liste mit Blocknummern freier Blöcke):
(1, 3, 8)

Durch einen Systemabsturz ist das Dateisystem inkonsistent geworden. Sie wissen lediglich, dass alle I-Nodes korrekt durch mindestens einen Verzeichniseintrag referenziert werden.

a) Führen Sie einen Algorithmus aus, mit dem widersprüchliche Block-Informationen innerhalb der I-Node-Liste sowie zwischen I-Node-Liste und Freiliste ermittelt werden können.

Aufbau einer Tabelle durch Inspektion aller I-Nodes und der Freiliste (ein Eintrag bedeutet die Anzahl an Verweisen auf diesen Block in der jeweiligen Liste):

0	1	2	3	4	5	6	7	8	9	Blocknummer
1	0	1	2	1	1	1	1	0	0	Block in I-Nodeliste referenziert
0	1	0	1	0	0	0	0	1	0	Block in Freiliste referenziert

b) Geben Sie alle gefundenen widersprüchlichen Informationen an und erklären Sie jeweils, wie Sie diesen Zustand bereinigen würden!

- Block 3: Doppelter Datenblock und gleichzeitig in Freiliste
  → Block 3 in freien Block 1 kopieren und einem der beiden I-Nodes statt Block 3 zuweisen
  → Block 3 und 1 aus Freiliste entfernen
- Block 9: Vermisster Block → Block 9 in Freiliste eintragen