Lehrstuhl für Automatisierung und Informationssysteme
Prof. Dr.-Ing. B. Vogel-Heuser

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Wintersemester 20[illegible]7/18

09.03.20[illegible]

Bitte legen Sie Ihren Lichtbildausweis bereit.

Sie haben für die Bearbeitung der Klau[illegible] 120 Minuten Zeit.

Diese Prüfung enthält [illegible] nummerierte Seiten inkl. Deckblatt.

Bitte prüfen Sie die Vollständigkeit Ihres Exemplars!

Bitte nicht r[illegible] oder grün schreibenden Stiften oder Bleistift ausfüllen!

Aufgabe	Erreichte Punkte
1	
2	
3	
4	
5	
6	
∑G	
7	
8	
9	
10	
11	
∑BS	
12	
13	
14	
15	
16	
17	
∑MSE	
18	
19	
20	
21	
22	
∑C	
∑	

Vorname Nachname _____ Matrikelnummer _____

Aufgabe G: Grundlagen

Aufgabe G:
39 Punkte

1. IEEE 754 Gleitkommazahlen

Rechnen Sie die gegebene Gleitkommazahl (angelehnt an die IEEE 754 Darstellung) in eine Dezimalzahl um.

Hinweis: Ergebnisse und Nebenrechnungen außerhalb der dafür vorgesehenen Textblöcke werden nicht bewertet!

0	1	0	0	1	1	1	0	0	1
V	e (4 Bit)				M (5 Bit)				

Vorzeichen

V= „+“ oder „positiv“

Bias und biased Exponent

B= $2^{4-1}-1=7$ **e =** 9

Exponent

E = $e - B = 9 - 7 = 2$

Mantisse (Dualzahl und Denormalisiert)

M_2= $(1{,}11001)_2 * 2^2 = (111{,}001)_2$

Vollständige Dezimalzahl Z (inkl. Vorzeichen)

Z= +7,125

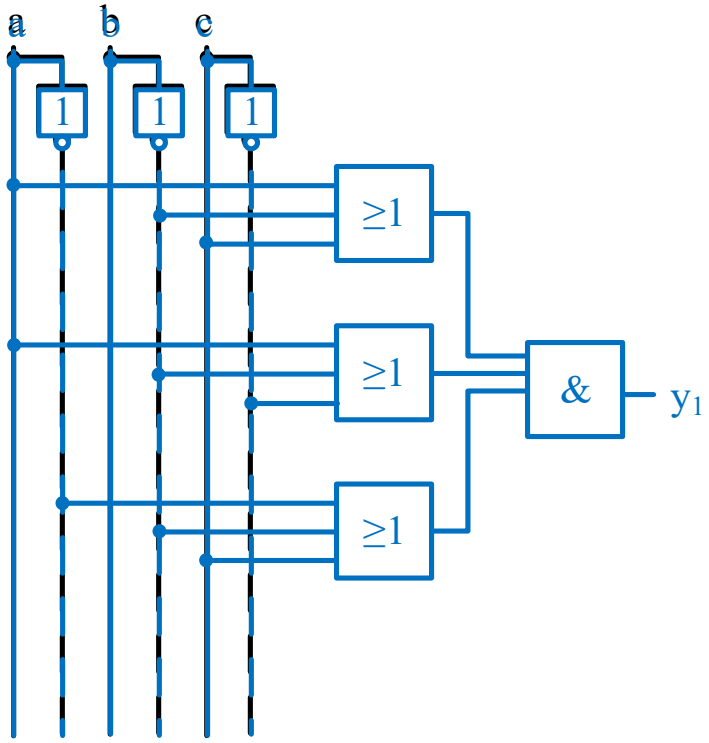
Vorname Nachname

Matrikelnummer

2. Logische Schaltungen und Schaltbilder

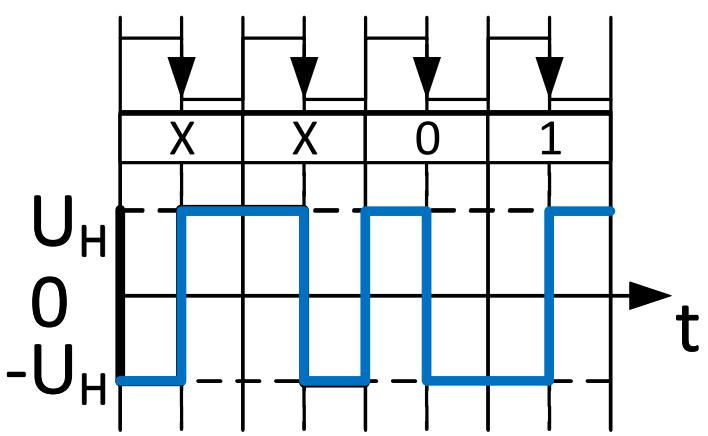
Sie sind zuständig für den Schaltungsentwurf. Ihnen wurde die angegebene Wahrheitstabelle übergeben. Erstellen Sie eine *graphische Schaltung in Normalform* (KNF / DNF). Erstellen sie diejenige Normalform, die am *wenigsten Schaltglieder* erfordert.

a	b	c	y_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



3. Leitungscodes

Die folgende *Binärfolge* soll mit Hilfe eines seriellen Bussystems übertragen werden. Hierfür müssen die Daten *serialisiert* werden. Verwenden Sie dazu den *Differential-Manchester-Code*. *Hinweis:* Beachten Sie das angegebene Taktsignal.



Vorname Nachname | Matrikelnummer

4. Normalformen und Minimierung

Gegeben sind folgende KV-Diagramme.

☐ **Dieses KV-Diagramm werten**

	a	a	ā	ā
b	X	1	1	0
b̄	X	0	1	X
	c̄		c	c̄

Bild G-4.1: *KV-Diagramm 1*

☐ **Dieses KV-Diagramm werten**

	a	a	ā	ā
b	X	1	1	0
b̄	X	0	1	X
	c̄		c	c̄

Bild G-4.2: *KV-Diagramm 2*

Minimieren Sie das KV-Diagramm in Form der KNF *(Konjunktive Normalform)* durch Einrahmen (Schleifen) der entsprechenden Felder im oben dargestellten KV-Diagramm. Schreiben Sie die minimierte Funktion in boolescher Algebra in das Lösungsfeld unten auf. Die Felder mit y=„X“ sind don‘t care bits.

Hinweis: Das zweite abgebildete KV-Diagramm dient als Ersatz, falls Sie sich verzeichnen. Kennzeichnen Sie durch Ankreuzen im Feld „dieses KV-Diagramm werten", welches KV-Diagramm bewertet werden soll.

Formel: $y_{min} = c \wedge (\bar{a} \vee b)$

Vorname Nachname | Matrikelnummer

5. Flip-Flops

Gegeben ist die folgende Master-Slave Flip-Flop Schaltung (MS-FF)

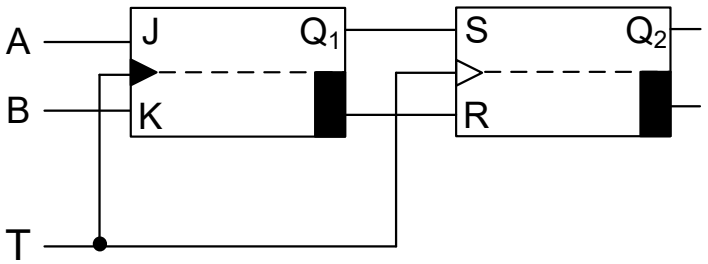
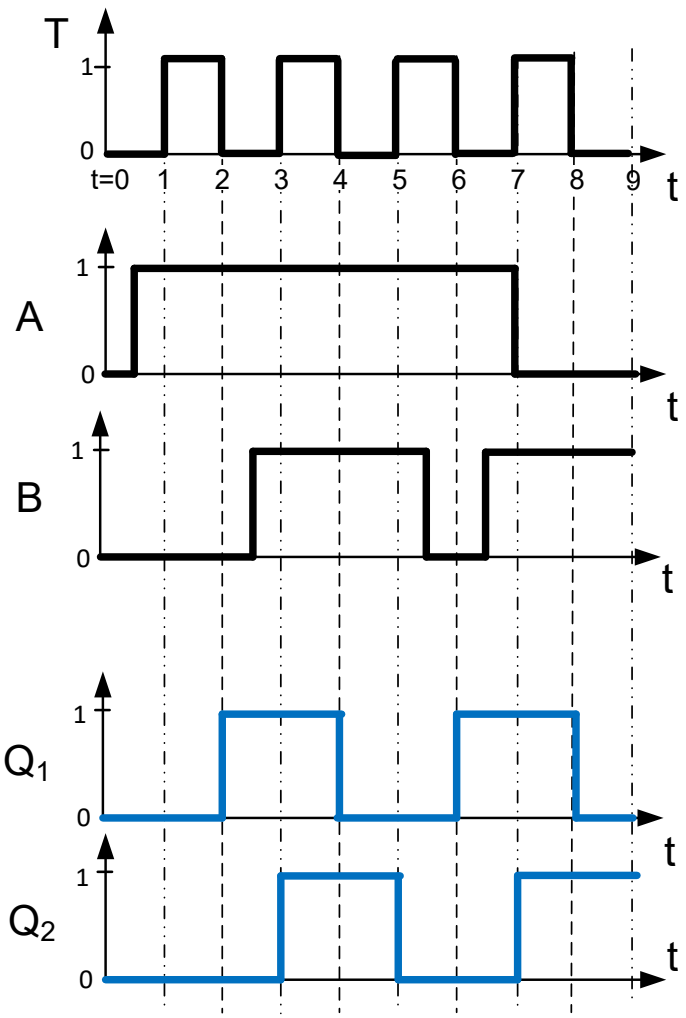


Bild G-5.1: *MS-FF*

Bei t = 0 sind die Flip-Flops in folgendem Zustand: $Q_1 = Q_2 = 0$.

Analysieren Sie die Schaltung für den Bereich $t = [0; 9[$, indem Sie für die Eingangssignale A, B und T die zeitlichen Verläufe für Q_1 und Q_2 in die vorgegebenen Koordinatensysteme eintragen.

Hinweis: Signallaufzeiten können bei der Analyse vernachlässigt werden.



Vorname Nachname | Matrikelnummer

6. MMIX-Rechner

Im Registerspeicher eines MMIX-Rechners befinden sich die in Bild G-6.2 gegebenen Werte. Es sollen nacheinander die zwei Befehle (Bild G-6.4) abgearbeitet und das Ergebnis in dem Registerspeicher (Bild G-6.2) bzw. Datenspeicher (Bild G-6.3) abgelegt werden.

	0x_0	0x_1	...	0x_4	0x_5	...
	0x_8	0x_9		0x_C	0x_D	
0x0_	TRAP	FCMP	...	FADD	FIX	...
	FLOT	FLOT I		SFLOT	SFLOT I	
0x1_	FMUL	FCMPE	...	FDIV	FSQRT	...
	MUL	MUL I		DIV	DIV I	
0x2_	ADD	ADD I	...	SUB	SUB I	...
	2ADDU	2ADDU I		8ADDU	8ADDU I	
...	...	...	...	...	...	...
0x8_	LDB	LDB I	...	LDW	LDW I	...
	LDT	LDT I		LDO	LDO I	
0x9_	LDSF	LDSF I	...	CSWAP	CSWAP I	...
	LDVTS	LDVTS I		PREGO	PREGO I	
0xA_	STB	STB I	...	STW	STW I	...
	STT	STT I		STO	STO I	
...	...	...	...	...	...	...
0xE_	SETH	SETMH	...	INCH	INCMH	...
	ORH	ORMH		ANDNH	ANDNMH	
0xF_	JMP	JMP B	...	GETA	GETA B	...

Bild G-6.1: *MMIX-Code-Tabelle*

Registerspeicher	
Adresse	Wert vor Befehlsausführung
...	...
$0x87	0x00 00 00 00 00 01 B0 3C
$0x88	0x00 00 00 00 DE AC 10 CF
$0x89	0x00 00 00 00 00 00 00 0E
$0x8A	0x00 00 00 00 00 00 61 FE
...	...

Bild G-6.2: *Registerspeicher*

Datenspeicher	
Adresse	Wert
...	...
0x00 00 00 00 00 00 61 FF	0xF0
0x00 00 00 00 00 00 62 00	0x01
0x00 00 00 00 00 00 62 01	0xDA
0x00 00 00 00 00 00 62 02	0x53
0x00 00 00 00 00 00 62 03	0x1B
0x00 00 00 00 00 00 62 04	0x00
0x00 00 00 00 00 00 62 05	0xB0
...	...

Bild G-6.3: *Datenspeicher*

Nr.	Maschinensprache	Assemblersprache	Befehlsbeschreibung
1	0x24 89 87 89	?	?
2	?	?	M_2[$0x8A+0x05]=$0x87

Bild G-6.4: *Befehle in Maschinensprache, Assemblersprache und Befehlsbeschreibung*

Bearbeiten Sie nun folgende Fragen (nächste Seite) zu den Befehlen und zu den Änderungen, die sich durch die Befehle ergeben.

Vorname Nachname _____ Matrikelnummer _____

Befehl Zeile 1:

Unterstrichenes ist wichtig der Rest ist Herleitung / Erklärung, z.B. 0x oder die führenden Nullen kann man auch weglassen.

a) Geben Sie die Formulierung in Assemblersprache für den Befehl Nr. 1 in Bild G-6.4 an.

0x24 89 87 89 → SUB $0x89, $0x87, $0x89

b) Wie lautet die Befehlsbeschreibung des in Nr. 1 in Bild G-6.4 angegebenen Befehls?
Hinweis: Ein Beispiel für eine Befehlsbeschreibung ist in Bild G-6.4, Zeile Nr. 2 gegeben.

$0x89 = $0x87 - $0x89, oder „Speichert die Differenz des Registerinhalts $0x87 und $0x89 in Register $0x89“

c) Geben Sie die Adresse und den Wert der durch den Befehl Nr. 1 in Bild G-6.4 geänderten Registerspeicherzelle an!

Schriftliche Subtraktion:

```
  00 01 B0 3C
- 00 00 00 0E
       1
```

Adresse: Adresse: 0x89

Wert: Wert: 0x0..00 00 01 B0 2E

Befehl Zeile 2:

d) Geben Sie die Formulierung in Maschinensprache für Befehl Nr. 2 in Bild G-6.4 an.
Hinweis: Ein Beispiel für einen Befehl in Maschinensprache ist in Bild G-6.4, Zeile Nr. 1 gegeben.

M_2[$0x8A+0x05]=$0x87 → STWI $0x87, $0x8A, 0x05 → 0xA5 87 8A 05

e) Geben Sie Adressen und Werte der durch Befehl Nr. 2 in Bild G-6.4 geänderten Datenspeicherzelle(n) an!

Adresse		Wert
0x00 …	62 02	B0
0x00 …	62 03	3C
0x00 …		
0x00 …		

$0x8A = 0x0..61 FE → 0x 0x0..61 FE + 0x05 = 0x0..62 03 → nächste Datenspeicherzelle, die durch 2 teilbar ist (Wyde!): 0x0..62 02 → Lese die 2 LSB aus Registerspeicher 0x87 und speichere den Inhalt in Datenspeicher ab 0x0..62 02

Vorname Nachname

Matrikelnummer

Aufgabe BS: Betriebssysteme

Aufgabe BS:
41 Punkte

7. Zeitparameter von Rechenzuständen

Gegeben sei das folgende Prozessmodell (Bild BS-7.1). Ordnen Sie jeweils die entsprechenden Zeitparameter den Lücken im Modell zu. Beantworten Sie außerdem die untenstehenden Fragen bezüglich Scheduling-Algorithmen.

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

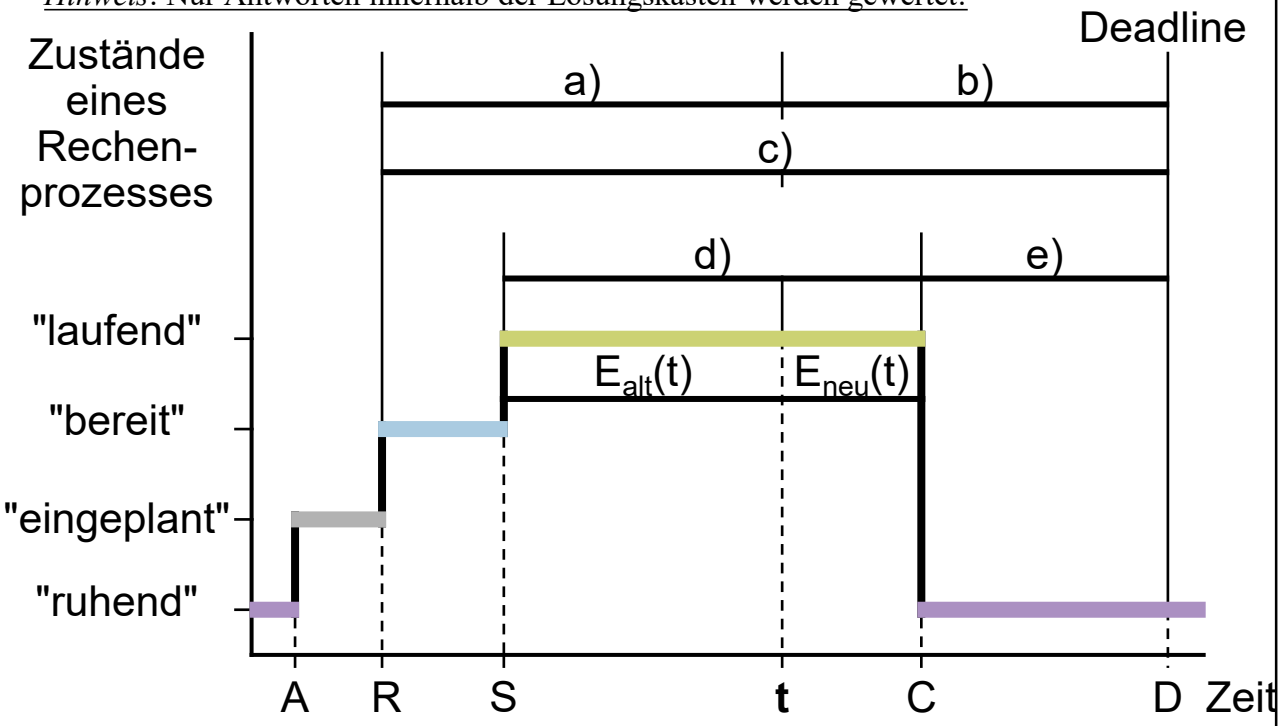


Bild BS-7.1: *Prozessmodell*

Wie lautet die Bezeichnung für Parameter c)? **Maximale Antwortzeitdauer**

Wie lautet die Bezeichnung für Parameter d)? **Maximale Ausführungszeitdauer**

Wie lautet die Formel zur Berechnung des Spielraums L? **L = D-E-S bzw. D-d)-S**

Bei welchem Scheduling-Algorithmus spielt der Parameter D die zentrale Rolle?

( ) LIFO ( ) LL (X) EDF ( ) Round Robin

Vorname Nachname | Matrikelnummer

8. Asynchrones Scheduling, präemptiv, RR

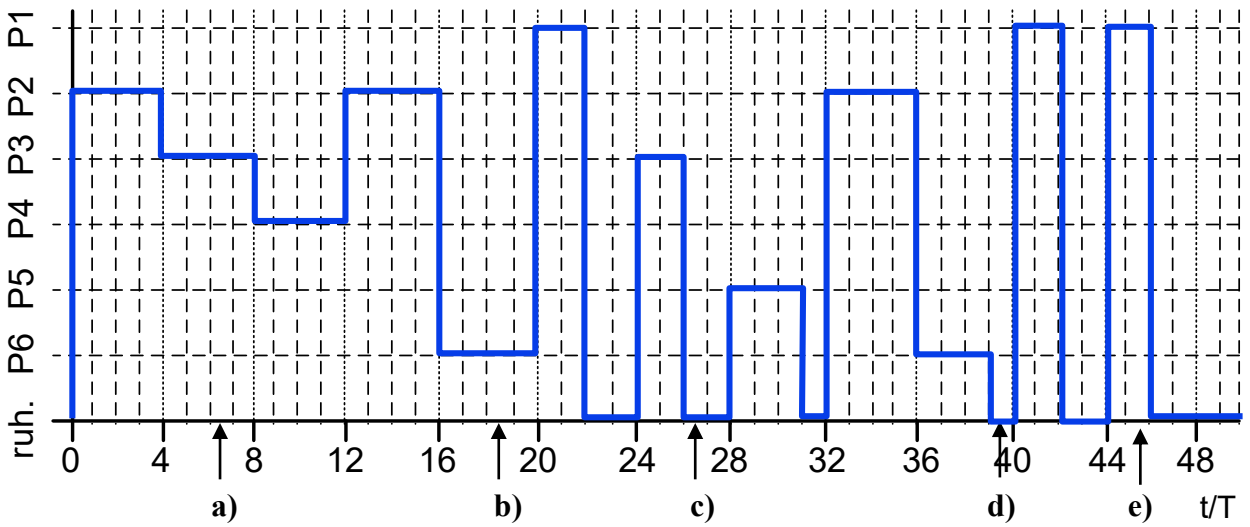
Gegeben seien die folgenden sechs Prozesse (Bild BS-8.1), welche jeweils ab dem Zeitpunkt „Start“ eingeplant werden sollen. Zur Abarbeitung eines Tasks wird die Rechenzeitspanne „Dauer“ benötigt. Periodische Tasks werden mit der Häufigkeit „Frequenz“ erneut aufgerufen. Erstellen Sie im untenstehenden Diagramm das präemptive Scheduling nach dem Schema „Round-Robin“ für den Zeitraum 0 bis 50s für einen Einkernprozessor. Treffen innerhalb eines Zeitschlitzes mehrere Tasks ein, beachten Sie die „Prioritäten“. Ein Zeitschlitz hat eine Größe von vier Sekunden. Tragen Sie die jeweils zu den gekennzeichneten Zeitpunkten aktiven Tasks ein.

Hinweis: Nur Antworten innerhalb des Lösungskastens werden gewertet!

	Priorität	Start	Dauer	Frequenz
P1	1 (hoch)	8 s	2 s	18 s
P2	2	0 s	12 s	einmalig
P3	3	2 s	6 s	einmalig

	Priorität	Start	Dauer	Frequenz
P4	4	3 s	4 s	einmalig
P5	5	10 s	3 s	einmalig
P6	6 (niedrig)	6 s	7 s	einmalig

Bild BS-8.1: *Taskspezifikation I*



a) P1 ( ), P2 ( ), P3 (x), P4 ( ), P5 ( ), P6 ( ), ruhend ( )

b) P1 ( ), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 (x), ruhend ( )

c) P1 ( ), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend (x)

d) P1 ( ), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend (x)

e) P1 (x), P2 ( ), P3 ( ), P4 ( ), P5 ( ), P6 ( ), ruhend ( )

Vorname Nachname | Matrikelnummer

9. Semaphoren

Gegeben seien die folgenden vier Tasks T1 bis T4 mit absteigender Priorität sowie die dazugehörigen Semaphoren S1 bis S4 (Bild BS-9.1). Die Startwerte der Semaphoren entnehmen Sie der Antworttabelle. Tragen Sie in der ersten Spalte der Antworttabelle den aktuell laufenden Task ein sowie im Rest der Zeile die Werte der Semaphoren nach Ausführung des jeweiligen Tasks.

T1	T2	T3	T4
P(S1)	P(S2)	P(S3)	P(S1)
P(S1)		P(S3)	P(S4)
P(S3)			
P(S3)	…	…	…
…			V(S1)
V(S2)	V(S3)		V(S3)
V(S2)	V(S4)	V(S1)	V(S3)

Bild BS-9.1: *Semaphorenzuweisung*

Task	S1	S2	S3	S4
-	0	2	0	0
T2	**0**	**1**	**1**	**1**
T2	**0**	**0**	**2**	**2**
T3	**1**	**0**	**0**	**2**
T4	**1**	**0**	**2**	**1**
T3	**2**	**0**	**0**	**1**
T4	**2**	**0**	**2**	**0**
T1	**0**	**2**	**0**	**0**

Vorname Nachname Matrikelnummer

10. Echtzeitbetriebssysteme

Im Folgenden (Bild BS-10.1) ist ein unvollständiges „erweitertes Taskzustandsdiagramm von RTOS-UH“ gegeben.

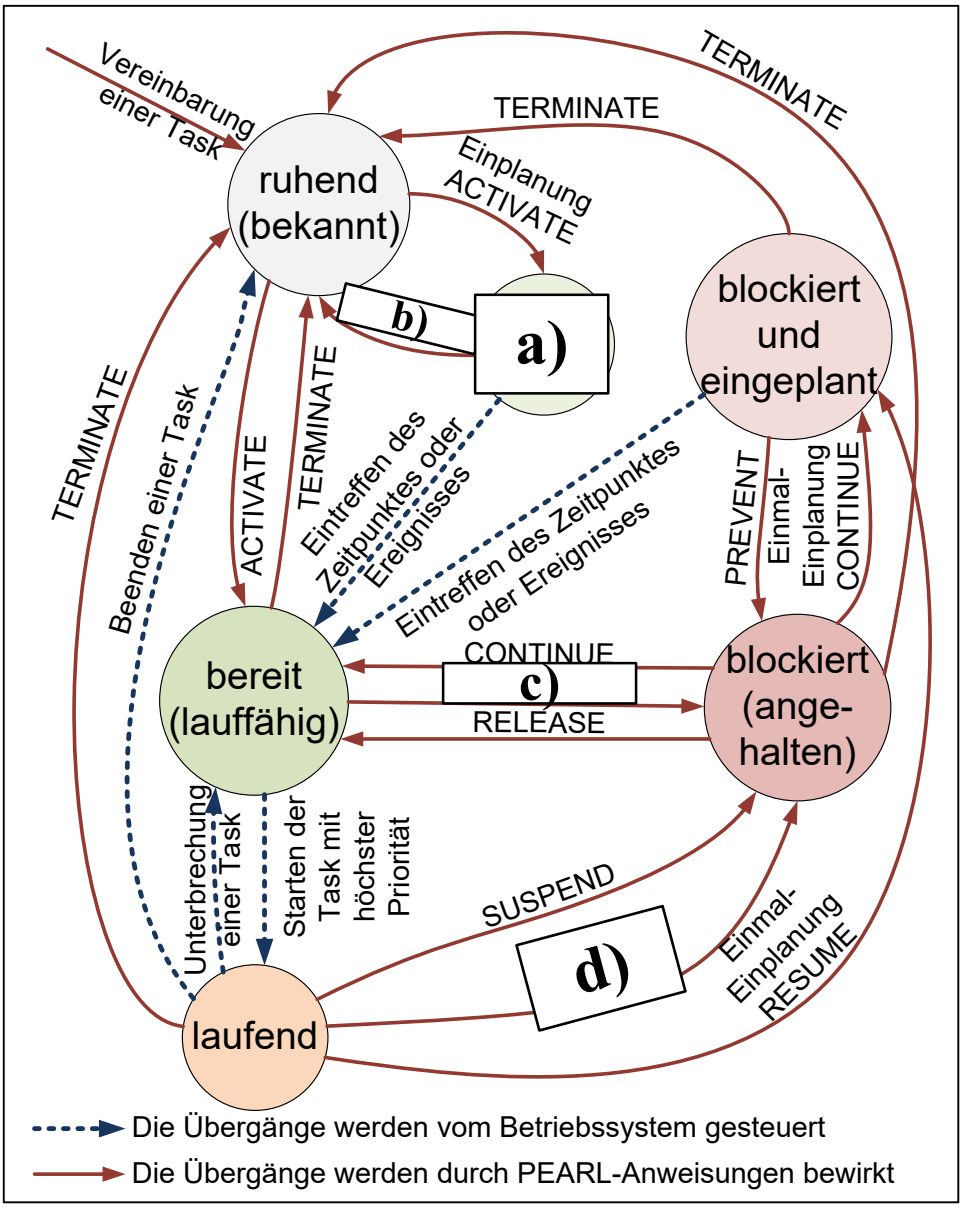


Bild BS-10.1: *Erweitertes Taskzustandsdiagramm von RTOS-UH*

Bezeichnen Sie die im Diagramm mit Buchstaben markierten Lücken:

a) **Eingeplant**

b) **PREVENT**

c) **SUSPEND**

d) **Erfolgloses REQUEST**

Vorname Nachname Matrikelnummer

11. IEC 61131-3: Funktionsbausteinsprache (FBS)

Die Tür einer S-Bahn soll über vier Tasten (*Tür_öffnen_außen, Tür_öffnen_innen, Nothalt, Tür_verriegeln*) sowie einen Sensor (*Zug_in_Bewegung*) gesteuert werden. Der Sollzustand der Tür wird über einen Ausgang (*Tür_offen*) angesteuert.

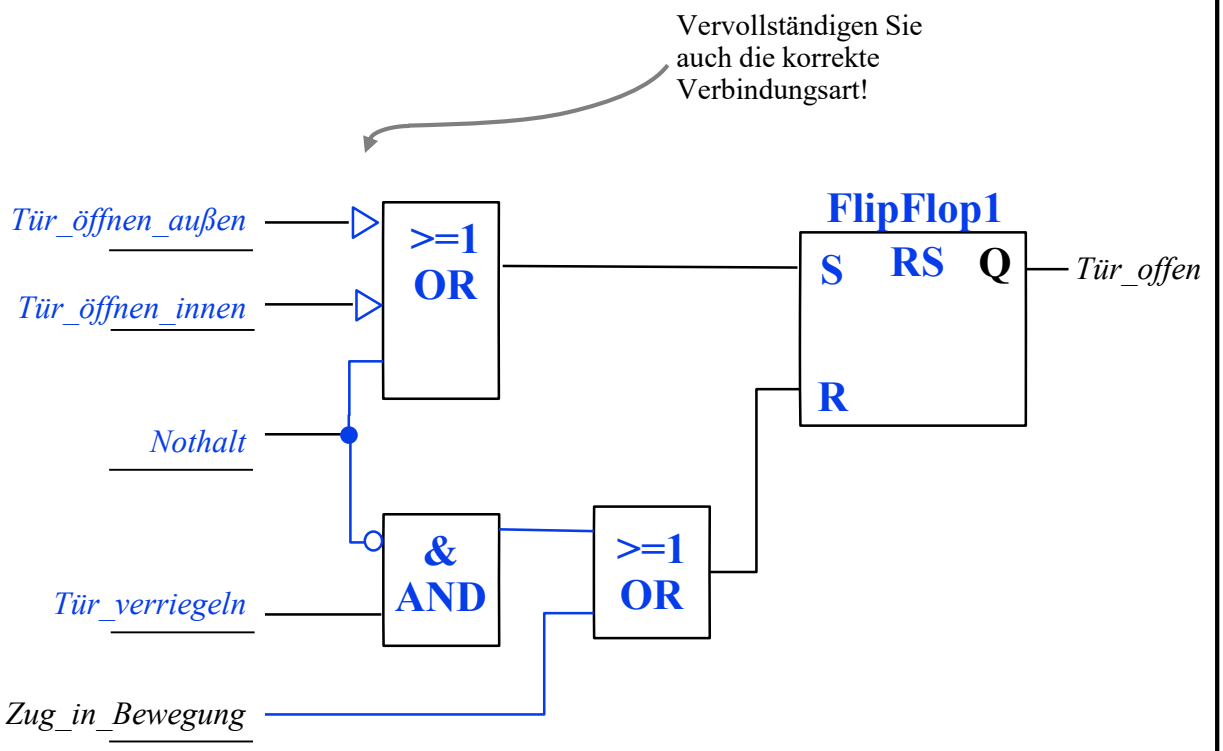
Wechselt das Signal einer der Tasten der Fahrgäste innen oder außen an der Tür von 0 auf 1 soll sich die Tür öffnen. Wird eine der Tasten länger gedrückt gehalten oder ist verklemmt, könnte die Steuerung beschädigt werden und muss somit bei der Signalauswertung unterbunden werden.

Der Zugführer verfügt über den Schalter *Tür_verriegeln*, mit deren Hilfe er die Taster der Fahrgäste übersteuern kann und die Tür schließt so lange der Zustand dieses Schalters 1 ist.

Für den Notfall existiert der *Nothalt* -Schalter, der im Zustand 1 die Taster der Fahrgäste und den Schalter des Zugführers übersteuert und die Tür öffnet sobald der Zug zum Stillstand gekommen ist.

Abschließend dürfen die Türen nur geöffnet sein, so lange sich der Zug nicht bewegt. Sobald der Bewegungssensor *Zug_in_Bewegung* ausgelöst wird und auf 1 wechselt, muss die Tür in jedem Fall geschlossen werden.

Hinweis: Signalverzögerungen im System sind zu vernachlässigen



Vorname Nachname _____ Matrikelnummer _____

Aufgabe MSE: Modellierung und Softwareentwicklung

Aufgabe MSE:
64 Punkte

12. Automaten

a) Gegeben sei der nachfolgende Automat. Der Automat befindet sich aktuell im Zustand *s2*.
Leiten Sie eine Übersicht der Übergänge ab, indem sie die untenstehende Tabelle vervollständigen (Zustand und Ausgabe)!

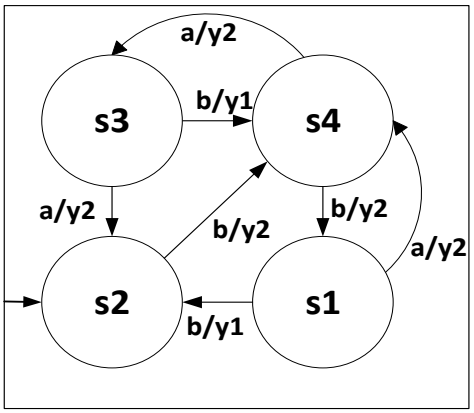


Bild MSE-12.1: *Automat*

T	s1	s2	s3	s4
a	s4,y2	-	s2,y2	s3,y2
b	s2,y1	s4,y2	s4,y1	s1,y2

Welche Eingabe müssen Sie tätigen damit Sie die Ausgabesequenz
y2 y2 y2 y2 y2 y2 erhalten? Geben Sie eine mögliche Sequenz an.

b b a a a b / b a b b a a

Handelt es sich bei dem gegebenen Automaten um einen Moore- oder Mealy-Automaten?

Moore-Automat ( ) Mealy-Automat (x)

Benötigt ein Moore-Automat bei gleicher dargestellter Logik in der Regel mehr oder weniger Zustände als ein Mealy-Automat?

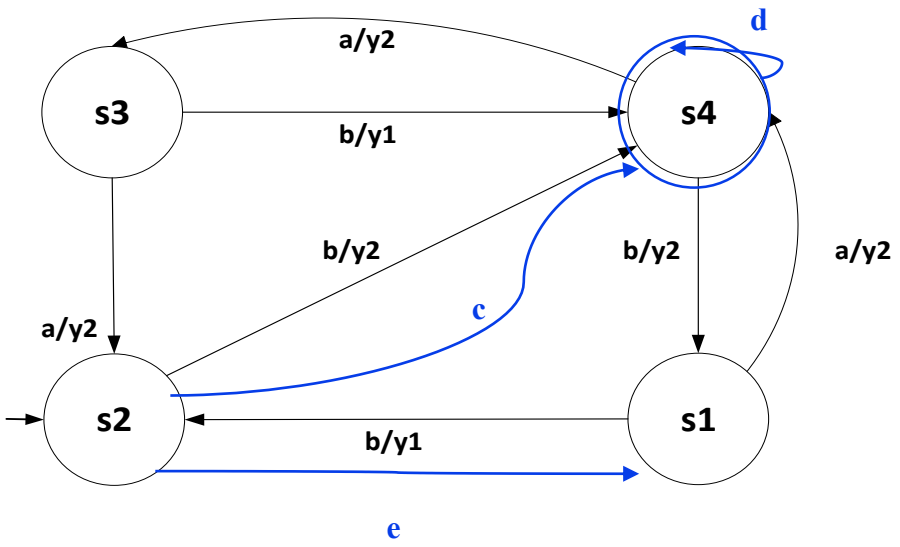
Mehr (x) Weniger ( )

Vorname Nachname | Matrikelnummer

b) Der Automat aus a) soll nun zu einem erkennenden Automaten erweitert werden:

- Der Anfangszustand *s2* bleibt bestehen.
- Der gültige Endzustand soll s4 sein.
- Ergänzen Sie jeweils maximal eine Transition, so dass die Eingabewörter „c“, „bbad“ und „ea“ gültig sind.

Vervollständigen Sie den Automaten gemäß der obigen Beschreibung!

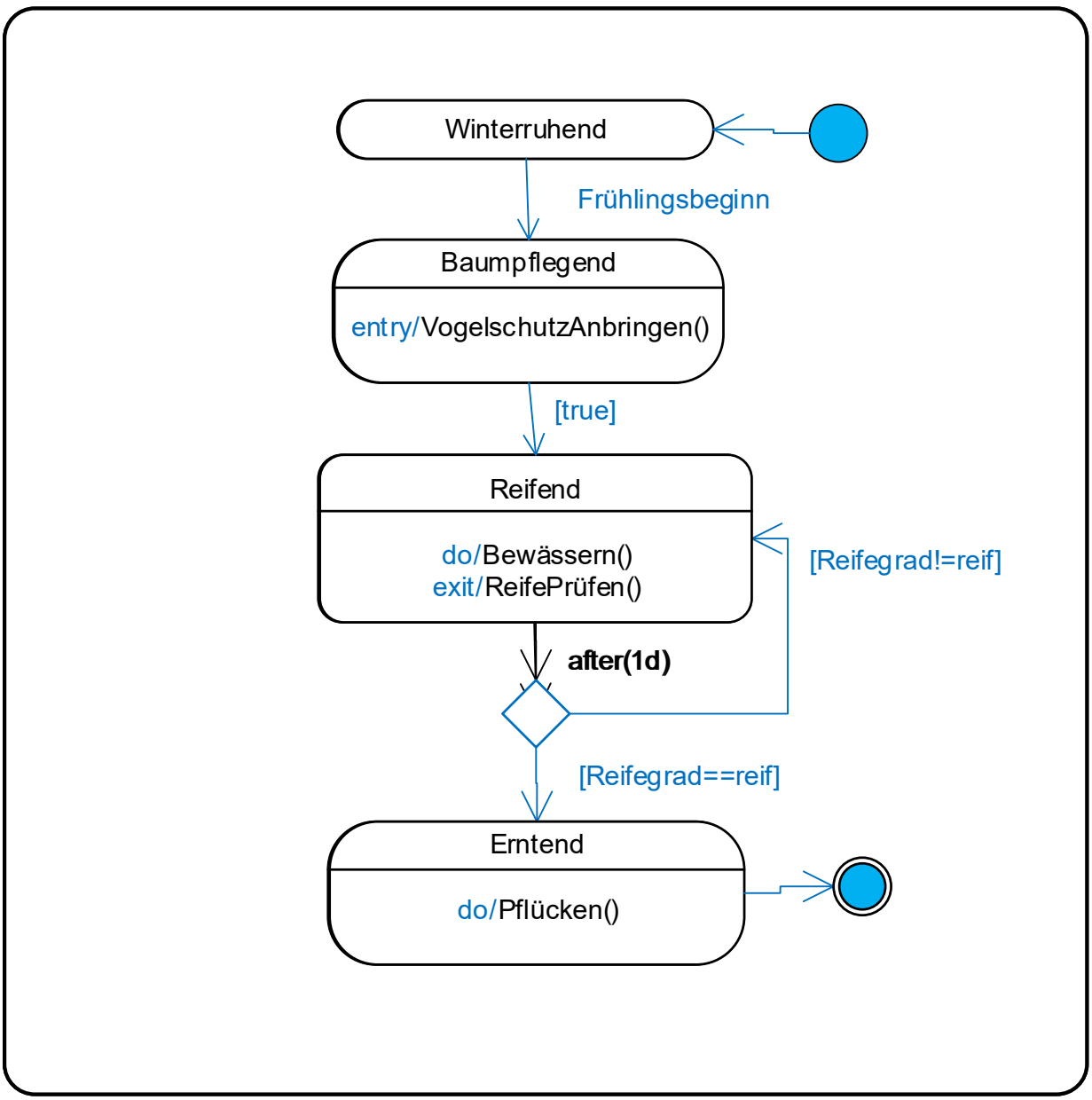


Vorname Nachname | Matrikelnummer

13. Zustandsdiagramm

Im folgenden soll der Anbau von Kirschen als Zustandsdiagramm modelliert werden. Die Anbauperiode beginnt zuerst im Zustand *Winterruhend*. Mittels des Triggers *Frühlingsbeginn* wird mit der Baumpflege (Zustand *Baumpflegend*) begonnen, zu deren Beginn man einmalig den *VogelschutzAnbringen* muss. Nach Abschluss der Schutzmaßnahmen wird automatisch in den Zustand *Reifend* gewechselt. Hier muss der Bauer den Baum kontinuierlich *bewässern*. Nach Ablauf eines Tages muss er zudem die Kirschen auf *ReifePrüfen*. Entspricht die Variable *Reifegrad* dem Wert *reif*, so wird mit der Ernte begonnen. Andernfalls wird in den Zustand *Reifend* zurückgesprungen. Im Zustand *Erntend* soll kontinuierlich die Aktion *Pflücken* ausgeführt werden.

Vervollständigen Sie das Zustandsdiagramm. Fügen Sie Start- und Endzustand sowie die Ausführungszeitpunkte der Aktionen ein. Zeichnen Sie keine zusätzlichen Zustände ein.



Vorname Nachname | Matrikelnummer

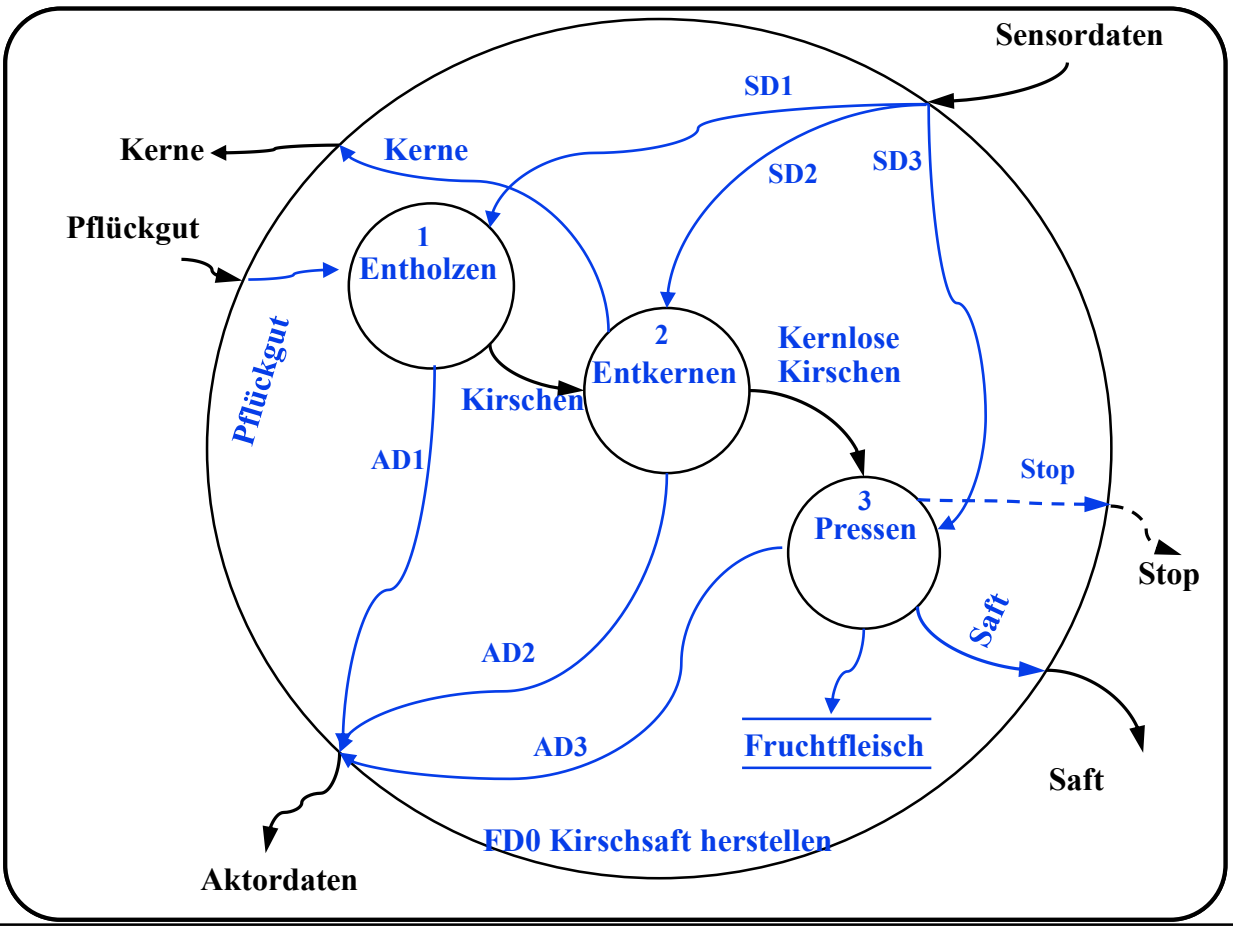
14. SA/RT: Flussdiagramm

Für die folgenden Teilaufgaben ist eine Maschine zur Herstellung von Kirschsaft gegeben, die aus drei Prozessschritten in jeweils spezialisierten Anlagenmodulen besteht.

Im Prozess *Entholzen* (Prozess 1) wird *Pflückgut* der Maschine zugeführt, welche die Stiele entfernt um *Kirschen* zu erhalten. Im Prozess *Entkernen* (Prozess 2) werden diese in *Kerne* und *kernlose Kirschen* getrennt. Im Prozess *Pressen* (Prozess 3) werden die *kernlosen Kirschen* gepresst, um *Fruchtfleisch* und *Saft* zu erhalten. Während der *Saft* abschließend an den nächsten Prozessschritt übergeben wird, verbleibt das *Fruchtfleisch* in einem internen Speicher. Um bei einem Problem in der Presse nachfolgende Maschinen stoppen zu können, verfügt die Presse über die Möglichkeit, ein *Stop*-Kommando zu senden.

Modellieren Sie den Prozess *Kirschsaft herstellen* (Prozess 0) mittels Strukturierter Analyse / Real-Time (SA/RT) in einem *Flussdiagramm* (FD0). Identifizieren Sie hierzu alle Subprozesse, Daten-, und Steuerflüsse und tragen Sie diese mit Bezeichnung ein. Beachten Sie, dass Sensor- und Aktordaten eines Prozesses mit SD*Prozessnummer* und AD*Prozessnummer* (z.B. SD1 oder AD3) zusammengefasst werden. Beachten Sie außerdem folgende Informationen:

Flüsse Material: Pflückgut, Kirschen, Kerne, Kernlose Kirschen, Saft, Fruchtfleisch	**Flüsse Sensordaten:** je nach Prozess: SD*Prozessnummer*, also z.B. SD2 bei „Entkernen“.
Steuerflüsse: Stop	**Flüsse Aktordaten:** Wie Sensordaten nur AD*Prozessnummer*



Vorname Nachname | Matrikelnummer

15. Antwortzeitspezifikation: Timing-Diagramm

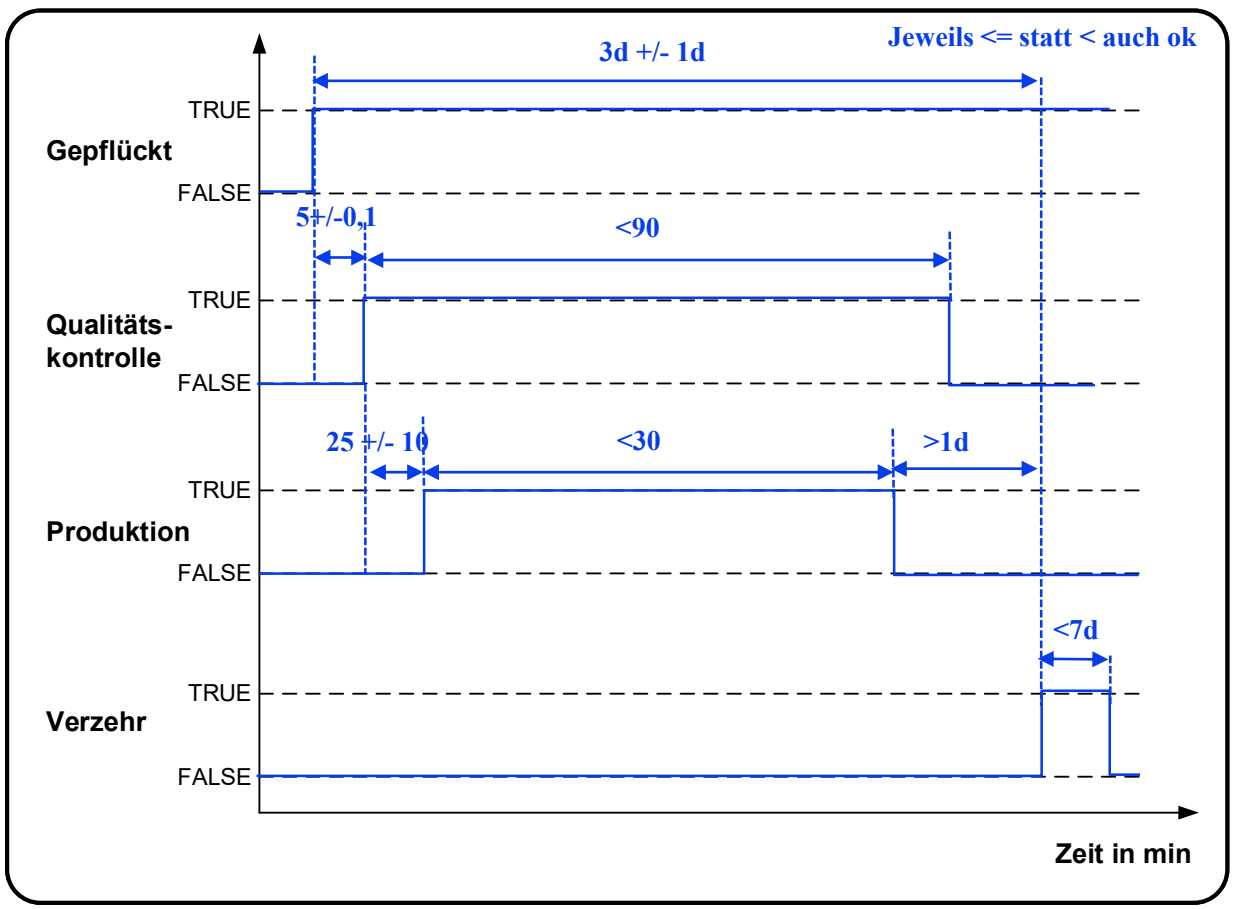
Nachfolgend soll die Herstellung sowie der Verzehr von Kirschsaft in Form eines Timing-Diagramms erstellt werden, um sicherzustellen, dass der Prozess korrekt durchgeführt wird.

Die Zustände der Prozesse können jeweils *TRUE* (z.B. Qualitätskontrolle aktiv) oder *FALSE* (z.B. Qualitätskontrolle abgeschlossen) sein.

Ergänzen Sie das Timing-Diagramm gemäß folgender Angaben (Werteverläufe und Zeitangaben):

- Sobald alle Kirschen gepflückt sind (*Gepflückt=TRUE*), muss nach 5 Minuten mit einer Toleranz von 0,1 Minuten durch Entnahme einer Probe mit der Qualitätskontrolle begonnen werden. Die anschließende Dauer zur Kontrolle beträgt höchstens 90 Minuten.
- Innerhalb von 15 bis 35 Minuten nach Beginn der Qualitätskontrolle, wird mit der Produktion begonnen. Dieser Vorgang darf maximal 30 Minuten benötigen. Der fertig produzierte Saft darf frühestens nach einem Tag verzehrt werden.
- Der gesamte Vorgang vom Pflücken bis Beginn des Verzehrs soll 2 bis 4 Tage dauern. Der anschließende Verzehr muss innerhalb von 7 Tagen abgeschlossen sein.

Eine maßstabsgetreue Darstellung der Zeiten ist nicht notwendig. Kürzen Sie falls nötig die Einheit Tag mit „d“ ab.



Vorname Nachname _____ Matrikelnummer _____

16. SA/RT Architekturdiagramme

Beurteilen Sie die Behauptungen auf ihre Richtigkeit hin.

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

	wahr	*falsch*
Im Architekturflussdiagramm werden die einzelnen Signale einer Anlage konkreten Bussystemen zugeordnet	( )	(x)
Das Architecture Dictionary kann verwendet werden, um die Echtzeitfähigkeit der geplanten Bussysteme zu beurteilen	(x)	( )
Die Nachverfolgbarkeitsmatrix stellt die Zuordnung von Architekturkomponenten zu Anforderungsmodellkomponenten dar, an denen sie beteiligt sind	(x)	( )

17. Bussysteme

Beurteilen Sie die Behauptungen auf ihre Richtigkeit hin.

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

	wahr	*falsch*
Bei FlexRay handelt es sich um ein dezentral gesteuertes Buszugriffsverfahren	(x)	( )
CSMA/CD eignet sich zum priorisierten Versand wichtiger Nachrichten	( )	(x)
Der CAN-Bus setzt CSMA/CA als Buszugriffsverfahren ein	(x)	( )

Vorname Nachname _____ Matrikelnummer _____

Aufgabe C: C-Programmierung

Aufgabe 18: 10 Punkte

18. C: Datentypen und Boolsche Algebra

a) Datentypen

Definieren Sie die Datentypen der folgenden Variablen so, dass so wenig Speicher wie möglich benötigt wird. Die Variablen sollen in einem System zur Verwaltung eines Zoos verwendet werden.

```
Variable gewicht      Gewicht des Zootieres als ganze Zahl in
                      Kilogramm. (gewicht > 3000kg).
Variable alter        Alter des Zootieres als ganze Zahl
                      (ältestes Tier < 200 Jahre)
Variable avgAlter     Durchschnittsalter aller Tiere im Zoo als
                      Gleitkommazahl
```

	char	int	double	float
gewicht	( )	(x)	( )	( )
alter	(x)	( )	( )	( )
avgAlter	( )	( )	( )	(x)

Welchen Datentyp hat das Ergebnis folgender Berechnung: `(int * char) / (float)`

( ) char ( ) int ( ) double (x) float

b) Ein- und Ausgabe

Füllen Sie die folgenden Ein- und Ausgabebefehle sowie die Formatstrings aus.

b.1 Ausgabe einer Gleitkommazahl mit 1 Nachkommastelle und 5 Zeichen Breite.

printf (" %5.1f ", fZahl);

b.2 Einlesen eines Strings von maximal 40 Zeichen von der Standardeingabe.

fgets(array, 40 , stdin);

Vorname Nachname _____ Matrikelnummer _____

c) Boolsche Algebra

Bestimmen Sie das Ergebnis der nachfolgend angegebenen Ausdrücke im Dezimalsystem. Gegeben sind folgende Variablen:

```
int a = 3;
int b = 18;
int c = 1;
int* d = &a;
```

Nach jedem Ausdruck werden die Variablen auf die oben genannten Werte zurückgesetzt.

c.1	`b / a + c * *d + c`	**10**
c.2	`((a & b) << 2) >= (a * c) << 2`	**00**

d) Boolsche Ausdrücke

Schreiben Sie jeweils einen boolschen Ausdruck, der die Aussagen der gegebenen textuellen Beschreibungen wiedergibt.
Die Variablen `int iZahl1`, `int iZahl2` und `int iZahl3` sind bereits definiert.

d.1 iZahl1 ist größer als 2 oder iZahl2 und iZahl3 sind nicht größer als iZahl1

```
iZahl1 > 2 || !(iZahl2 > iZahl1 && iZahl3 > iZahl1)
```

d.2 iZahl1 ist gerade und iZahl2 ist ungerade und größer oder gleich als iZahl3

```
iZahl1 % 2 == 0 && iZahl2 % 2 != 0 && iZahl2 >= iZahl3
```

Vorname Nachname | Matrikelnummer

19. Kontrollstrukturen

Aufgabe 19: 10 Punkte

Sie sollen ein Programm erstellen, das die Lieblingsfuttersorte eines Zootieres berechnet. Hierzu steht Ihnen ein Array *verzehrtesFutter* zur Verfügung, in dem gespeichert ist wie viele Tonnen eine von drei Tierarten (Giraffe, Kamel und Tapir) von jeder der drei Sorten Futter gefressen hat.

Berechnen Sie die Futtersorte, die jede Tierart bevorzugt, und speichern Sie den Index dieser Sorte im Array *lieblingsSorte*. Geben Sie zum Schluss die bevorzugte Sorte jeder Tierart sowie die verzehrte Menge in Tonnen mit einer Nachkommastelle aus. Wählen Sie die korrekte Antwortmöglichkeit für die Lücken in Abbildung C-19.1 aus den Alternativen, die in Abbildung C-19.2 auf der nachfolgenden Seite angegeben sind, aus.

```
        (1)
_____________________
int main() {
//X-Richtung: Futtersorten; Y-Richtung: Tierarten
  float verzehrtesFutter [][3] = {{5.5, 1, 1},
                                  {5.1, 6.2, 1},
                                  {5, 2.5, 6.4}};
  int lieblingsSorte[3] = {0, 0, 0};
  int futter = 0, tier = 0; float temp = 0;

  for (_____(2)_____ ) {
    for (futter = 0; futter < 3; futter++) {
      if (_____(3)_____) {
        lieblingsSorte[tier] =___(4)___;
        temp = verzehrtesFutter[tier][futter];
      }
    }
  }

  for (tier = 0; tier < 3; tier++) {
    ____(5)____ {
      ____(6)____
          printf ("Die Giraffe ");___(7)___
      ____(6)____
          printf ("Das Kamel ");  ___(7)___
      ____(6)____
          printf ("Der Tapir ");  ___(7)___
      }
      printf("bevorzugt Sorte %i und fraß___(8)___t\n",
      lieblingsSorte[tier] + 1,
      ____(9)_____
    }
  ____(10)____
}
```

Abbildung C-19.1: Sourcecode zur Berechnung der bevorzugten Futtersorte der im Zoo vorhandenen Tierarten.

Vorname Nachname _____ Matrikelnummer _____

1. Kreuzen Sie in der folgenden Abbildung C-19.2 die korrekten Codefragmente für die in Abbildung C-19.1 angegebenen Lücken an (nur Einfachantwort möglich).

Lücke Nr. 1 in Abbildung C-19.1

(1)
```
( ) #include stdio.h     ( ) #include <stdio>     (x) #include <stdio.h>
( ) #define <stdlib.h>   ( ) #include stdlib      ( ) #define <io.h>
```

Lücke Nr. 2 in Abbildung C-19.1

(2)
```
(x) tier = 0; tier < 3; tier++
( ) tier = 0; tier <= 3; tier++
( ) futter = 0; futter < 3; futter++
( ) tier = 1; tier < 3; tier++
( ) tier = 0; tier < 3; tier
( ) tier = 0; tier < 3; futter++
```

Lücke Nr. 3 in Abbildung C-19.1

(3)
```
( ) verzehrtesFutter[futter][tier] > temp
( ) verzehrtesFutter[tier + 1][futter + 1] > temp
( ) verzehrtesFutter[tier][futter] < temp
( ) verzehrtesFutter[tier][futter] > temp++
( ) verzehrtesFutter[tier][futter] > temp - futter
(x) verzehrtesFutter[tier][futter] > temp
```

Lücke Nr. 4 in Abbildung C-19.1

(4)
```
( ) tier                ( ) tier++              ( ) futter + 1
( ) futter++            (x) futter              ( ) futer--
```

Lücke Nr. 5 in Abbildung C-19.1

(5)
```
( ) switch (futter)     ( ) switch tier         ( ) switch(lieblingsSorte)
(x) switch (tier)       ( ) switch(tier++)      ( ) if (tier)
```

Lücke Nr. 6 in Abbildung C-19.1

(6)
```
( ) case 1:   …   case 2:    …    case 3:    …
(x) case 0:   …   case 1:    …    case 2:    …
( ) case "0": …   case "2":  …    case "3":  …
( ) case: 0   …   case: 1    …    case: 2    …
( ) case 1    …   case 2     …    case 3     …
( ) case 0    …   case 1     …    case 2     …
```

Lücke Nr. 7 in Abbildung C-19.1

(7)
```
( ) switch;             ( ) continue;           (x) break;
( ) default;            ( ) stop;               ( ) return;
```

Lücke Nr. 8 in Abbildung C-19.1

(8)
```
( ) %d                  ( ) %.1i                (x) %.1f
( ) %.2f                ( ) %1.f                ( ) %f
```

Lücke Nr. 9 in Abbildung C-19.1

(9)
```
( ) verzehrtesFutter[tier,lieblingsSorte[tier]]
( ) verzehrtesFutter[futter][lieblingsSorte[futter]]
( ) verzehrtesFutter[futter][lieblingsSorte[temp]]
(x) verzehrtesFutter[tier][lieblingsSorte[tier]]
( ) verzehrtesFutter[tier][lieblingsSorte[tier++]]
( ) verzehrtesFutter[futter][lieblingsSorte[tier]]
```

Lücke Nr. 10 in Abbildung C-19.1

(10)
```
( ) return              ( ) break 0;            ( ) continue 1;
( ) continue;           ( ) break;              (x) return 0;
```

Abbildung C-19.2: Codefragmente für Abbildung C-19.1.

Vorname Nachname _____ Matrikelnummer _____

20. Objektorientierte Programmierung

Aufgabe 20: 25 Punkte

a) Grundlagen & Konzepte

Folgend werden grundlegende Konzepte der objektorientierten Programmierung abgefragt. Bitte wählen Sie aus den Antwortalternativen die eine korrekte Alternative aus (nur Einfachnennung möglich).

a.1 Was trifft bei einer Vererbung nicht zu?
( ) Die Unterklasse kann Methoden der Oberklasse überschreiben.
( ) Eine Instanz der Unterklasse ist immer auch Instanz der Oberklasse.
(x) Die Oberklasse ist spezialisierter als die Unterklasse.
( ) Die Unterklasse steht in einer is-a Beziehung zur Oberklasse.
( ) Die Unterklasse ist spezialisierter als die Oberklasse.
( ) Die Unterklasse kann Methoden und Attribute hinzufügen.

a.2 Wie wird es bezeichnet wenn eine Klasse eine andere Klasse enthält?

( ) Kapselung	( ) Assoziation	( ) Implementierung
( ) Instanziierung	(x) Aggregation	( ) Vererbung

a.3 Instanzen welcher Klassen können auf private Attribute zugreifen?
( ) Instanzen aller anderen Klassen.
( ) Instanzen von aggregierten Klassen.
( ) Keine
( ) Instanzen von assoziierten Klassen.
(x) Instanzen der Klasse in der das Attribut definiert wurde.
( ) Instanzen von Unterklassen oder Klasse.

a.4 Was beschreibt das Prinzip der Datenkapselung?
( ) Der Zustand eines Objekts kann nicht verändert werden.
( ) Der Zustand eines Objekts darf nur von außen verändert werden.
( ) Alle Zustandsdaten werden in einem eigenen Objekt zusammengefasst.
( ) Ein Objekt muss alle seine Attribute im Konstruktor initialisieren.
(x) Der Zustand eines Objektes kann nur über Methoden verändert werden.
( ) Es kann nicht auf den Zustand eines Objekts zugegriffen werden.

a.5 In welcher Beziehung steht eine Telefonnummer zu einer Klasse Telefonbuch?
( ) Realisierung
( ) In keiner Beziehung
( ) Instanziierung
( ) Vererbung
(x) Aggregation
( ) Telefonnummer implementiert Telefonbuch

a.6 Was beschreiben die Attribute eines Objekts?

( ) Schnittstelle	(x) Zustand	( ) Instanz
( ) Kommunikation	( ) Abstraktion	( ) Verhalten

Vorname Nachname Matrikelnummer

b) Modellierung mit UML

Sie werden beauftragt eine Software zur Verwaltung des Tierbestands in einem Zoo zu entwickeln. Da sie sich mit den Eigenschaften von exotischen Tierarten nicht auskennen, erfragen Sie die Informationen in einem Interview mit einem Tierpfleger. Ein Ausschnitt der Interviewergebnisse, den Sie in ein Klassendiagramm überführen sollen, ist in Abbildung C-20.1 gegeben.

- Jedes exotische Tier hat einen Namen. Zudem muss eine Funktion vorhanden sein, die das Lieblingsfutter des Tieres als Zeichenkette zurückgibt.
- Derzeit werden zwei konkrete exotische Tierarten (Zebras und Elefanten) gehalten.
- Die Anzahl Streifen eines Zebras ist zur Identifikation der Tiere notwendig und muss als Ganzzahl gespeichert werden.
- Eine Herde besteht aus exotischen Tieren und ist einem Gehege zugeordnet.
- Einer Herde soll ein neues Tier mittels einer Funktion hinzugefügt werden können.

Abbildung C-20.1: Ergebnisse des Interviews mit dem Tierpfleger.

Beachten Sie folgende Konventionen: Attribute sollen mit privater Sichtbarkeit und Methoden mit öffentlicher Sichtbarkeit versehen werden. Verwenden sie, außer wenn dies explizit angegeben ist, ungerichtete Assoziationen.

Füllen Sie die Lücken im folgenden Klassendiagramm (Abbildung C-20.2). Die auszufüllenden Lücken sind mit Zahlen markiert. Die Namen der Variablen und Methoden sind in dieser Teilaufgabe nicht relevant. Achten Sie bei der Wahl der Datentypen auf möglichst geringen Speicherverbrauch.

Lücken **zwischen** Klassen erfordern das Einzeichnen von Beziehungen, Lücken **innerhalb** von Klassen erfordern das Einfüllen von Attributen, Methoden oder Klassennamen.

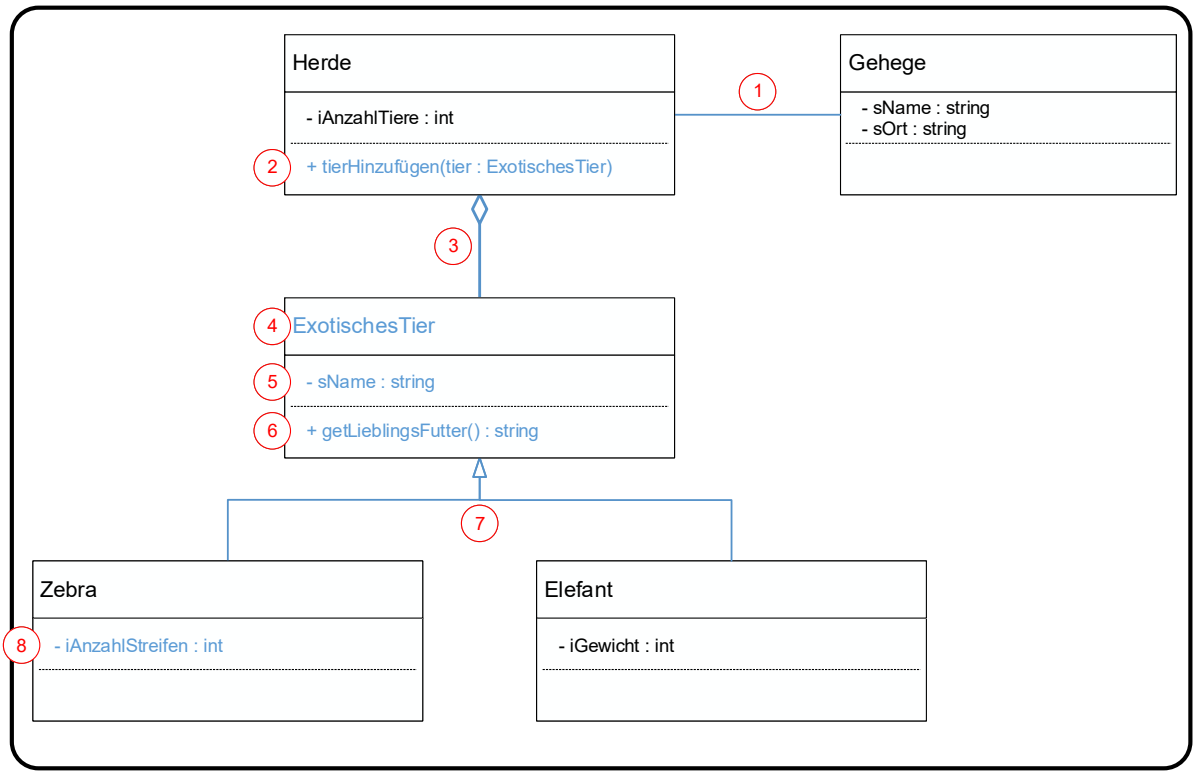


Abbildung C-20.2: Klassendiagramm der Verwaltungssoftware.

Vorname Nachname | Matrikelnummer

c) Vom Code zum Klassendiagramm

Sie haben Programmcode der Zooverwaltungssoftware erhalten. Um die Struktur der Software zu verstehen, möchten Sie ein Klassendiagramm des Codeausschnitts in Abbildung C-20.3 erstellen.

```
class Mitarbeiter
{
  public:
    Mitarbeiter(string name);

  private:
    string sName;
};

class Tierpfleger:public Mitarbeiter
{
  public:
    Tierpfleger();
    void setLieblingsTier(&ExotischesTier tier);

  private:
     ExotischesTier* lieblingsTier;
};

class ExotischesTier
{
  private:
    float fGewicht;

};
```

Abbildung C-20.3: Programmcode zur Tierpflegerverwaltung.

Ein Grundgerüst des Klassendiagramms ist in Abbildung C-20.4 vorgegeben. Die auszufüllenden Lücken sind mit Zahlen markiert.

Lücken **zwischen** Klassen erfordern das Einzeichnen von Beziehungen, Lücken **innerhalb** von Klassen erfordern das Einfüllen von Attributen, Methoden oder Klassennamen.

Achten Sie beim Ausfüllen auf die Datentypen und Namen der Variablen, der Parameter und der Rückgabewerte der Funktionen, die Sichtbarkeiten von Variablen und Methoden und die in Teilaufgabe **b)** genannten Konventionen.

Vorname Nachname

Matrikelnummer

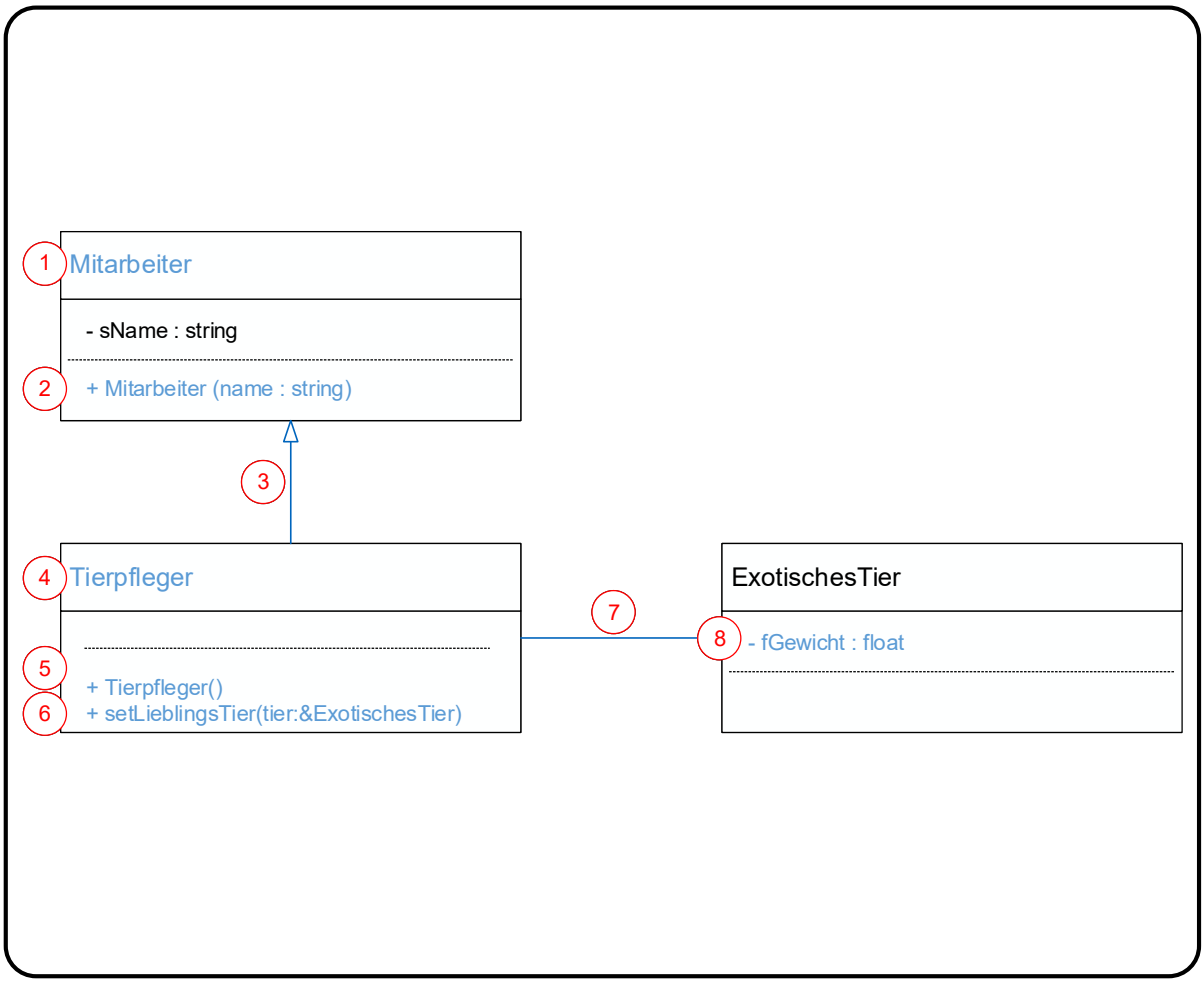


Abbildung C-20.4: Klassendiagramm des Programmcode zur Tierpflegerverwaltung (siehe Abbildung C-20.3 auf der vorherigen Seite).

Vorname Nachname | Matrikelnummer

21. Anlagen/Zustandsautomat

Aufgabe 21:
25 Punkte

Ihre Aufgabe ist es, eine sogenannte Giraffenwaschanlage zur Wäsche von Giraffen zu automatisieren. Nach der Bestellung der Hardwarekomponenten und deren Aufbau fehlt der Steuerungscode, welcher nun geschrieben werden muss. Zuvor muss das Verhalten der Anlage in einem Zustandsdiagramm abgebildet werden. Die Beschreibung der Funktionsweise der Anlage und Ihrer zu automatisierenden Komponenten folgt im nachfolgenden Abschnitt.

Die Anlage besteht aus zwei Bürsten *BOben* und *BSeite*, welche die Giraffen von oben (*BOben*) und seitlich (*BSeite*) reinigen sollen. Die Anlage soll sowohl von erwachsenen Giraffen (*große Giraffen*), als auch Jungtieren (*kleine Giraffen*) benutzt werden. Da kleine Giraffen zu klein sind, um die obere Bürste zu erreichen, sollen diese nur seitlich gebürstet werden. Für die Ermittlung der Höhe der Giraffen stehen zwei Lichtschranken zur Verfügung. *LSOben* detektiert nur *große Giraffen*, *LSUnten* ist bei *kleinen und großen Giraffen* aktiv. Zur Erfüllung der strikten Sicherheitsvorschriften beim Umgang mit exotischen Tieren wird weiterhin eine automatische Toranlage verbaut, welche den Bürstraum während Bürstvorgängen zum restlichen Gehege abschließen soll, damit nicht mehrere Giraffen gleichzeitig in der Anlage sind. Das Schiebetor kann auf- und zugefahren werden. Eine Lichtschranke (*LST*) im Torbereich detektiert, ob der Torbereich frei ist und das Tor geschlossen werden kann. Der gesamte Aufbau der Anlage ist in Abbildung C-21.1 zu sehen.

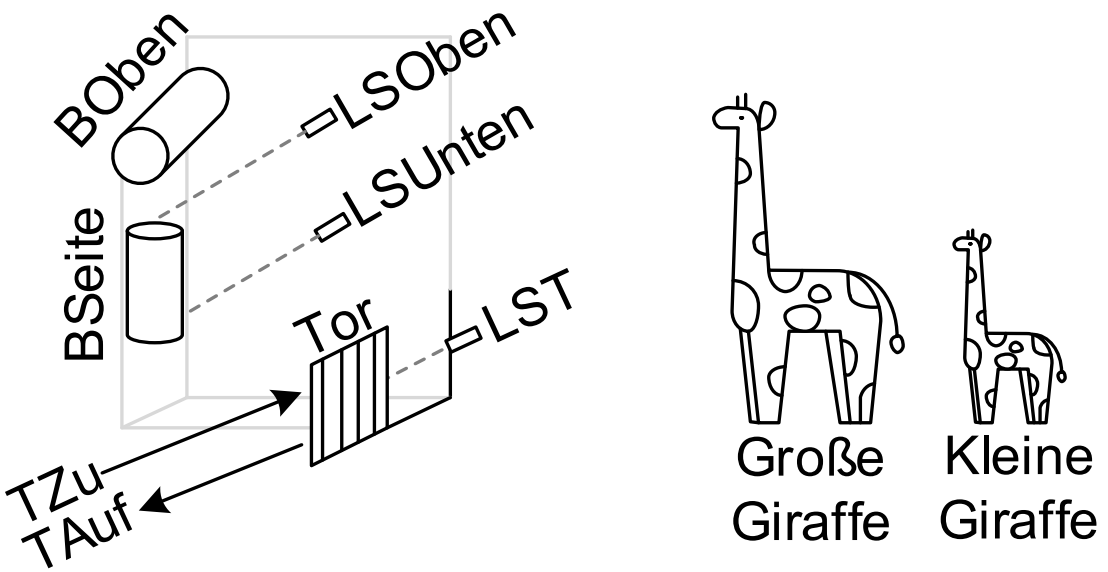


Abbildung C-21.1: Skizze der Giraffenwaschanlage mit schematischer Darstellung der Tiergrößen.

Vorname Nachname | Matrikelnummer

Die Liste an Ein- und Ausgängen ist wie folgt gegeben und im Header *„Buerste.h“* definiert:

Typ	Name	Beschreibung
AKTOREN	`a.BOben`	Startet (1) und stoppt (0) obere Bürste
	`a.BSeite`	Startet (1) und stoppt (0) seitliche Bürste
	`a.TAuf`	Öffnet (1) Tor oder stoppt Bewegung (0)
	`a.TZu`	Schließt (1) Tor oder stoppt Bewegung (0)
SENSOREN	`s.LSOben`	Liefert (1) falls große Giraffe in Anlage, sonst (0)
	`s.LSUnten`	Liefert (1) falls sich eine Giraffe in Anlage befindet, sonst (0)
	`s.LST`	Liefert (1) falls sich Tier im Torbereich aufhält, sonst (0)
	`s.TIstAuf`	Tor ist auf (1), sonst (0)
	`s.TIstZu`	Tor ist zu (1), sonst (0)
	`s.BSeiteFertig`	Zeigt das Ende des Bürstvorgangs der seitlichen Bürste *BSeite* an (1), sonst (0)
Variablen	`time`	Aktuelle Laufzeit des Programms in ms

Abbildung C-21.2: Sensor- und Aktorvariablen der Giraffenwaschanlage.

Zunächst soll die Anlage in einen definierten Ausgangszustand gebracht werden. Hierfür wird sichergestellt, dass beide Bürsten nicht aktiviert sind und das Tor aufgefahren ist. Wenn das Tor geöffnet ist, wird in den Wartemodus weitergeschaltet. In diesem Modus wartet die Waschanlage darauf, dass eine Giraffe die Anlage betritt. Wird mithilfe der Lichtschranken *LSUnten* die Anwesenheit einer beliebigen Giraffe detektiert, soll das Tor zugefahren werden. Es ist zu beachten, dass das Tor während des gesamten Bürstvorgangs aktiv geschlossen sein muss, der Aktor *TZu* also aktiviert bleiben muss, um ein Ausbrechen der Giraffe zu verhindern.
Tritt die Giraffe während des Schließvorgangs aus dem Waschbereich in die Lichtschranke *LST* muss der Vorgang sofort gestoppt werden und das Tor wieder geöffnet werden, um die Giraffe nicht einzuklemmen. Danach soll im Initalisierungszustand fortgefahren werden.
Sollte der Schließvorgang erfolgreich sein, die Giraffe also nicht *LST* blockiert haben, muss entschieden werden, welche Bürste aktiviert wird.
Im Falle einer großen Giraffe in der Anlage, wird zunächst nur die obere Bürste aktiviert. Nach 10 Sekunden Bürstvorgang wird diese Bürste gestoppt und die seitliche Bürste aktiviert. Befindet sich eine kleine Giraffe in der Anlage wird direkt die seitliche Bürste aktiviert und die obere bleibt deaktiviert.
Im Gegensatz zur oberen Bürste liefert die seitliche Bürste ein Signal nach Ende eines vorprogrammierten Bürstvorgangs zurück (*bSeiteFertig*) und wird nicht über eine Zeitbedingung kontrolliert. Nach Ende des Bürstens soll die Anlage wieder in den Initalisierungszustand versetzt werden und das Tor somit geöffnet werden.

Vorname Nachname | Matrikelnummer

a) Vervollständigen Sie das folgende, in Abbildung C-21.3 gegebene, Zustandsdiagramm entsprechend der Ablaufbeschreibung unter Berücksichtigung der in Abbildung C-21.2 gegebenen Signale. Füllen Sie hierzu die Bedingungen in den leeren Kästchen aus, beschreiben Sie die auszuführenden Aktionen im Falle von unterstrichenen Feldern innerhalb der Zustände und vervollständigen Sie die fehlenden Pfeile.

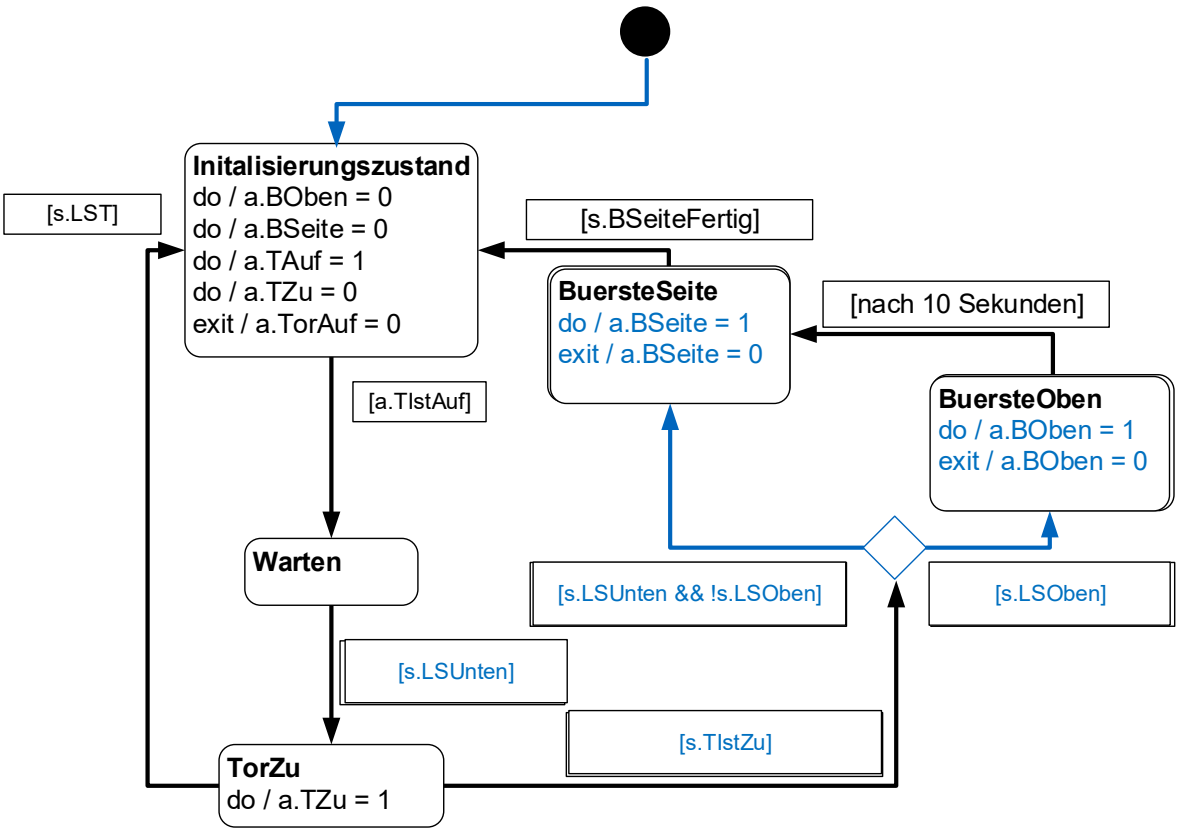


Abbildung C-21.3: Zustandsdiagramm der Giraffenwaschanlage.

Vorname Nachname | Matrikelnummer

b) Vervollständigen Sie nun den vorgegebenen Programmcode (siehe Abbildungen C-21.4 und C-21.5).

```
#include <stdio.h>
#include "buerste.h"

int schritt = 0;          // Schrittvariable
unsigned int t = 0;       // Timervariable

int main()
{
    switch (schritt     ) {
    case 0:               // Initial
        a.BOben = 0;
        a.BSeite = 0;
        a.TAuf = 1;
        a.TZu = 0;
        if (s.TIstAuf)    // Waechterbedingung
        {
            a.TAuf = 0;
            schritt = 1;
        }
        break;
    case 1:               // Warten
        if ( s.LSUnten                    )
        {
            schritt = 2;
        }
        break;
 case 2:                  // TorZu
        a.TZu = 1;
        if ( s.LST      ) { // Giraffe im Tor
            schritt = 0;
        }
        if (s.TIstZu)     // Tor geschlossen
        {
            if ( s.LSOBen           ) {
                schritt = 3;
            }
            else          {
                schritt = 4;
            }
        }
        break;
```

Abbildung C-21.4: Programmcode der Giraffenwaschanlage (Teil 1 von 2).

Vorname Nachname _____ Matrikelnummer _____

```
case 3:                                    // BuersteOben
        a.BOben = 1;
        if ( t == 0                       ) // Timer starten
        {
            t = time;
        }
        if ( t + 10000 < time             ) // Timer pruefen
        {
            a.BOben = 0;
            schritt = 4;
        }
        break;
    case 4: //BuersteSeite
        a.BSeite = 1;
        if (s.BSeiteFertig)
        {
            a.BSeite = 0;
            schritt = 0;
        }
        break;
    }
}
```

Abbildung C-21.5: Programmcode des Giraffenwaschanlage (Teil 2 von 2).

Vorname Nachname | Matrikelnummer

22. Erweiterte Datenstrukturen und FileIO

Aufgabe 22: 26 Punkte

Der Zoo benötigt eine Funktion um den Verzehr der Tiere in ihren Gehegen zu überwachen, die Kalkulation des Einkaufs von Futter zu erleichtern, sowie den „Vielfraß des Monats“ auf der Internetseite des Zoos zu küren. Sie arbeiten im Team, das für die Entwicklung dieser Software zuständig ist. Zunächst sollen geeignete Datenstrukturen zur Speicherung der Daten definiert werden. Es sollen zwei Strukturdatentypen definiert werden:

Der Typ `FUTTERBESTAND` speichert Messwerte der Füllstandssensoren an den Futterstellen. Hierzu wird zunächst der Zeitpunkt der Messung aufgenommen (`Timestamp`). Der Automat erzeugt für jeden Tag eine eigene Datei (`futterbestand.csv`, siehe Abbildung C-22.1), in welcher der Zeitpunkt einer Messung in der Form HHMM (HH = Stunde, MM = Minute), z.B. 1234 für 12:34 Uhr, angeben ist. Zu jedem Messwert wird eine eindeutige, ganzzahlige Identifikationsnummer für die Futterstelle gespeichert. Diese soll in einer Variablen mit dem Namen `Futterstelle` gespeichert werden. Zudem wird die Menge an verbrauchten Futter als Gleitkommazahl in der Variablen `Futtermenge` abgelegt.

In einer zweiten Datei (`futterstellen.csv`, siehe Abbildung C-22.2) werden die Identifikationsnummern der Futterstellen den Namen der Tiere die derzeit dort fressen zugeordnet. Zudem wird für die Auswertung die Gesamtmenge an verzehrtem Futter an der Futterstelle gespeichert. Hierzu soll ein Strukturdatentyp `FUTTERSTELLEN` definiert werden. Hierzu soll erneut die Nummer der Futterstelle gespeichert werden. Weiterhin besitzt jede Futterstelle ein zugeordnetes Tier, das im Feld Name als Zeichenkette gespeichert werden soll. Die erlaubte Länge des zu speichernden Tiernamens beträgt 20 ASCII-Zeichen. Die Einträge in `futterstellen.csv` sind nach aufsteigender Identifikationsnummer sortiert, wobei mit einer ID von 1 gestartet wird und keine Zahlen ausgelassen werden.

Vorname Nachname Matrikelnummer

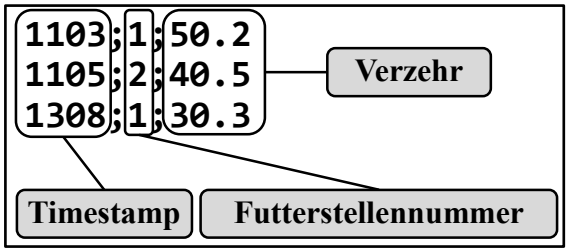


Abbildung C-22.1: Auszug aus der Datei `futterbestand.csv`.

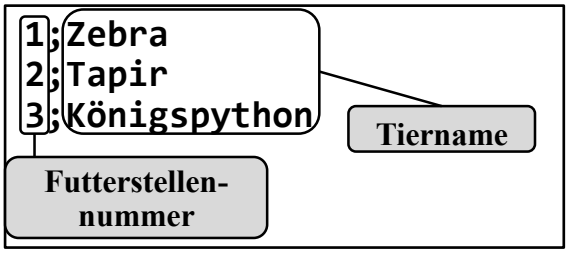


Abbildung C-22.2: Auszug aus der Datei `futterstellen.csv`.

a) Ergänzen Sie den untenstehenden Quelltext (Abbildung C-22.3) in der Headerdatei `datentypen.h` um die notwendigen Typdefinitionen.

```
// Futterbestand
typedef struct {
    unsigned int Timestamp;
    unsigned int Futterstelle;      // Futterstellennummer
    float Futtermenge;
}FUTTERBESTAND;

// Futterstellen
typedef struct {
    unsigned int Futterstelle;      //Futterstellennummer
    char Name[21];                  // Zugeordnetes Tier
    float GesamtVerzehr;           // Für Auswertung
}FUTTERSTELLEN;
```

Abbildung C-22.3: Typendefinitionen in der Headerdatei `datentypen.h`.

b) Die in der Headerdatei `datentypen.h` definierten Typen sollen nun in einem Programm zum Einlesen und Auswerten der einzelnen Dateien verwendet werden.

Vervollständigen Sie den in den Abbildungen C-22.5 und C-22.6 gegebenen Code. Der Programmentwurf ist in Abbildung C-22.4 als Nassi-Shneidermann-Diagramm gegeben.

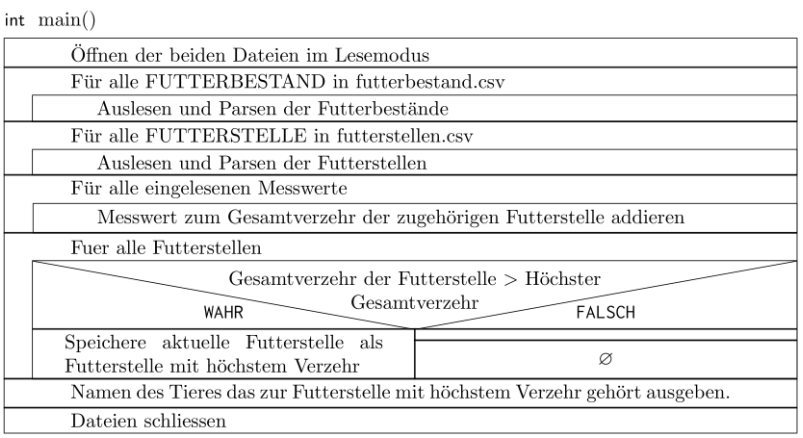


Abbildung C-22.4: Nassi-Shneiderman-Diagramm des Auswerteprogramms.

Vorname Nachname | Matrikelnummer

```
#include <stdio.h>
#include "datentypen.h"

//Konstanten mit vordefinierten Laengen
#define NMESSWERTE 10
#define NFUTTERSTELLEN 3
#define LENGTH 200

int main() {
  // Variablen und Pointer
  FUTTERBESTAND messwerte[NMESSWERTE];
  FUTTERSTELLEN futterstellen[NFUTTERSTELLEN];
  FUTTERBESTAND* pMW = messwerte;
  FUTTERSTELLEN* pFS = futterstellen;

  int i = 0;
  int j = 0;

  FILE*  a = fopen(futterbestand.csv, "rt" );
  FILE*  b = fopen(futterstellen.csv, "rt" );

  // Messwerte der Futterstellen einlesen
  for (i = 0; i < NMESSWERTE; i++)
  {
    fscanf  (a,"%i;%i;%f\n" ,  &pMW[i].Timestamp ,
      &pMW[i].Futterstelle , &pMW[i].Futtermenge );
  }

  // Futterstellen und zugeordnete Tiere einlesen
  for (i = 0; i < NFUTTERSTELLEN; i++)
  {
    fscanf (b, "%i;%s\n" , &pFS[i].Futterstelle ,
      &pFS[i].Name );
  }
```

Abbildung C-22.5: Programmcode des Auswerteprogramms (Teil 1 von 2).

Vorname Nachname Matrikelnummer

```
  FUTTERSTELLEN* hoechsterVerzehr = pFS;

  for (j = 0; j < NMESSWERTE; j++) {
    pFS[   pMW[j].Futterstelle - 1   ]. GesamtVerzehr
      +=   pMW[j].Futtermenge ;
  }

  for (i = 0; i < NFUTTERSTELLEN; i++) {
    if (pFS[i].GesamtVerzehr >
          hoechsterVerzehr -> GesamtVerzehr ) {
      hoechsterVerzehr   =  &pFS[i]  ;
    }
  }

  printf("Vielfrass des Monats ist ein:  %s  .",
     hoechsterVerzehr->Name     );

  fclose(a);      //Dateien schließen
  fclose(b);
  return 0;
}
```

Abbildung C-22.6: Programmcode des Auswerteprogramms (Teil 2 von 2).