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

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Sommersemester 2018

27.08.2018

Bitte legen Sie Ihren Lichtbildausweis bereit.

Sie haben für die Bearbeitung der Klausur 120 Minuten Zeit.

Diese Prüfung enthält 15 nummerierte Seiten inkl. Deckblatt.

Bitte prüfen Sie die Vollständigkeit Ihres Exemplars!

Bitte nicht rot- oder grün schreibenden Stiften oder Bleistift ausfüllen!

Aufgabe	Erreichte Punkte
1	2
2	7
3	2
4	2
5	6
6	7
7	7
8	16
∑GL	49
9	11
10	10
11	6
12	8
13	16
∑BS	51
14	14
15	12
16	7
17	4
18	7
∑MSE	44
19	11
20	12
21	23
22	24
23	26
∑C	96
∑	240

Aufgabe GL: Grundlagen

Aufgabe GL:
49 Punkte

1. Umrechnung zwischen Zahlensystemen

Nennen Sie die Basen der beiden Zahlensysteme, in die sich Zahlen des 4er-Systems besonders einfach umrechnen lassen.

2	16

2. IEEE 754 Gleitkommazahlen

Rechnen Sie die nachfolgende Dezimalzahl in eine Gleitkommazahl (angelehnt an die IEEE 754 Darstellung) mit folgender Formatierung um: $(-13{,}875)_{10}$ e (4bit) M (5bit)

V e (4 Bit) M (5 Bit)

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

Vorzeichen

V= 1

Mantisse (Binärzahl und Normalisiert)

M_2= $(1101{,}111)_2 = (1{,}101111 * 2^3)_2$

Exponent

E = 3

Bias und biased Exponent

B= $2^{(x-1)}-1 = 7$ **e =** $B + E = (10)_{10} = (1010)_2$

Vollständige Gleitkommazahl (10 bit)

1 1010 10111

Kann die gegebene Dezimalzahl bei gegebener Genauigkeit als Binärzahl exakt dargestellt werden?

	X
Ja	Nein

3. Quer- und Längsparität

Folgende Nachricht wurde mittels ungerade Parität gegen Übertragungsfehler geschützt. Finden und korrigieren Sie den/die minimalen Übertragungsfehler.

1	0	1	**1**
0	1 ~~0~~	1	**1**
1	1	1	**0**
1	**1**	**0**	**1**

4. Hamming-Distanz

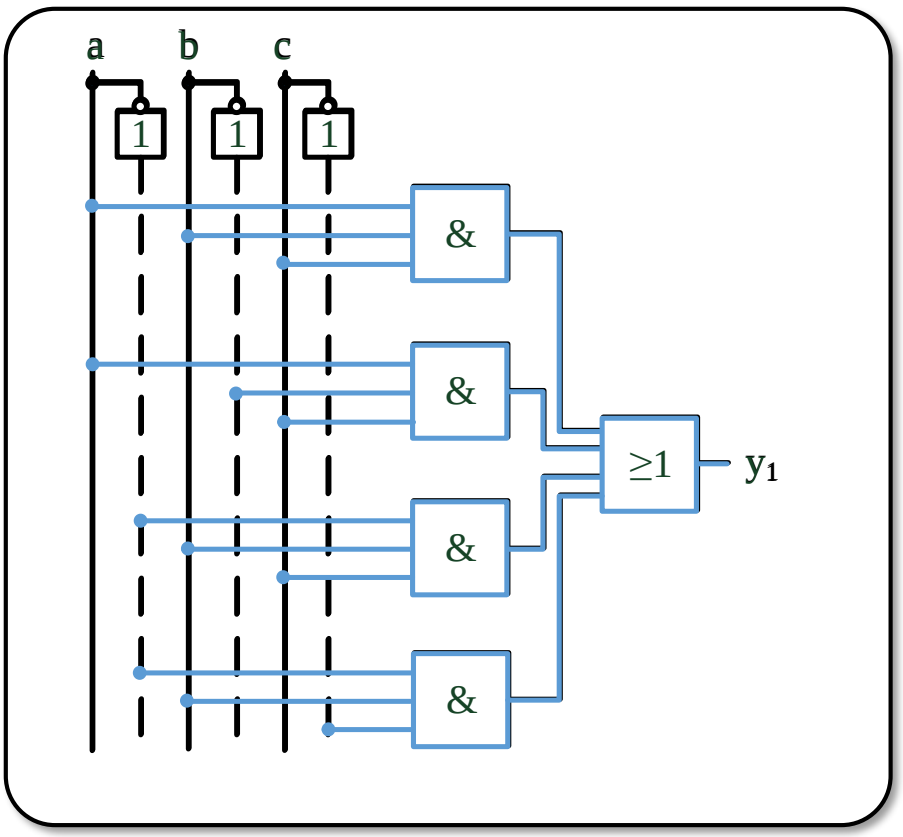
Gegeben sei eine Hamming-Distanz von 5. Wie viele Bitfehler können damit erkannt und wie viele behoben werden?

Erkennen: **h = e + 1 → e = 4**

Beheben: **h = 2f + 1 → f = 2**

5. Logische Schaltungen und Schaltbilder

Gegeben sei nebenstehende Wahrheitstabelle. Erstellen Sie das zugehörige Schaltbild der DNF.



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

Welche Schaltung ist hier dargestellt? Bitte kreuzen Sie den richtigen Fachbegriff an.

( ) XOR-Gatter für 3 Eingänge

( ) 2 Bit Syncron Vorwärtszähler

(x) Zweikanal-Multiplexer mit Selektionseingang „a“

( ) Dreikanal Demultiplexer

6. Normalformen und Minimierung

a)Welche Terme werden zur Bestimmung der KNF zusammengefasst?

( ) Minterme

(X) Maxterme

b) KV-Diagramme

Ermitteln Sie die minimierte KNF für die nebenstehende Wahrheitstabelle mittels eines KV-Diagramms.

Hinweis 1: Das Einzeichnen der Schleifen in das KV-Diagramm ist als Lösung ausreichend, die minimierte Formel muss nicht abgelesen werden.

Hinweis 2: 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.

Hinweis 3: „X" entspricht „don't care"-Einträgen

a	b	c	d	y_1
0	0	0	0	0
0	0	0	1	X
0	0	1	0	0
0	0	1	1	X
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	X
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	X
1	1	0	0	1
1	1	0	1	X
1	1	1	0	1
1	1	1	1	1

[X] Dieses KV-Diagramm werten

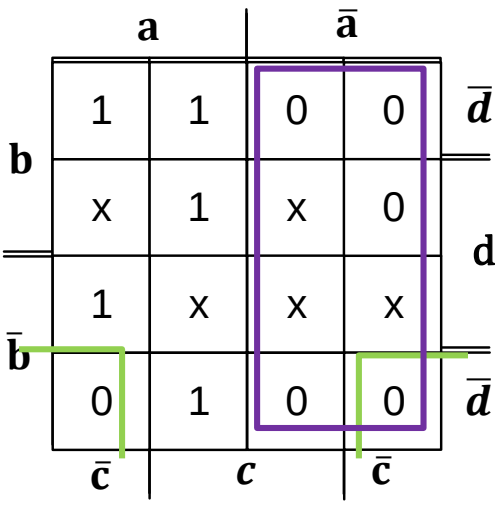


Bild G-6.1: *KV-Diagramm 1*

[ ] Dieses KV-Diagramm werten

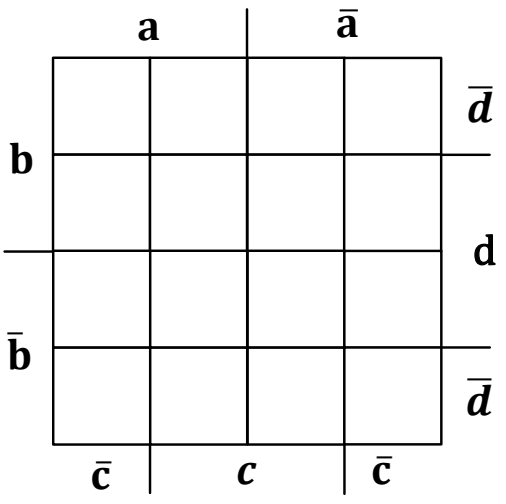


Bild G-6.2: *KV-Diagramm 2*

7. Flip-Flops

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

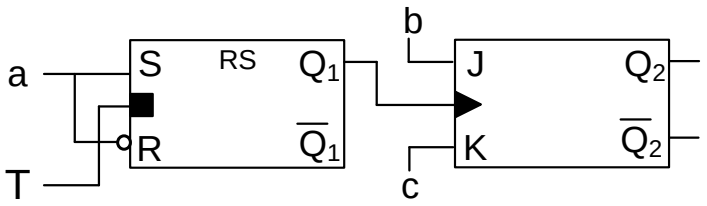
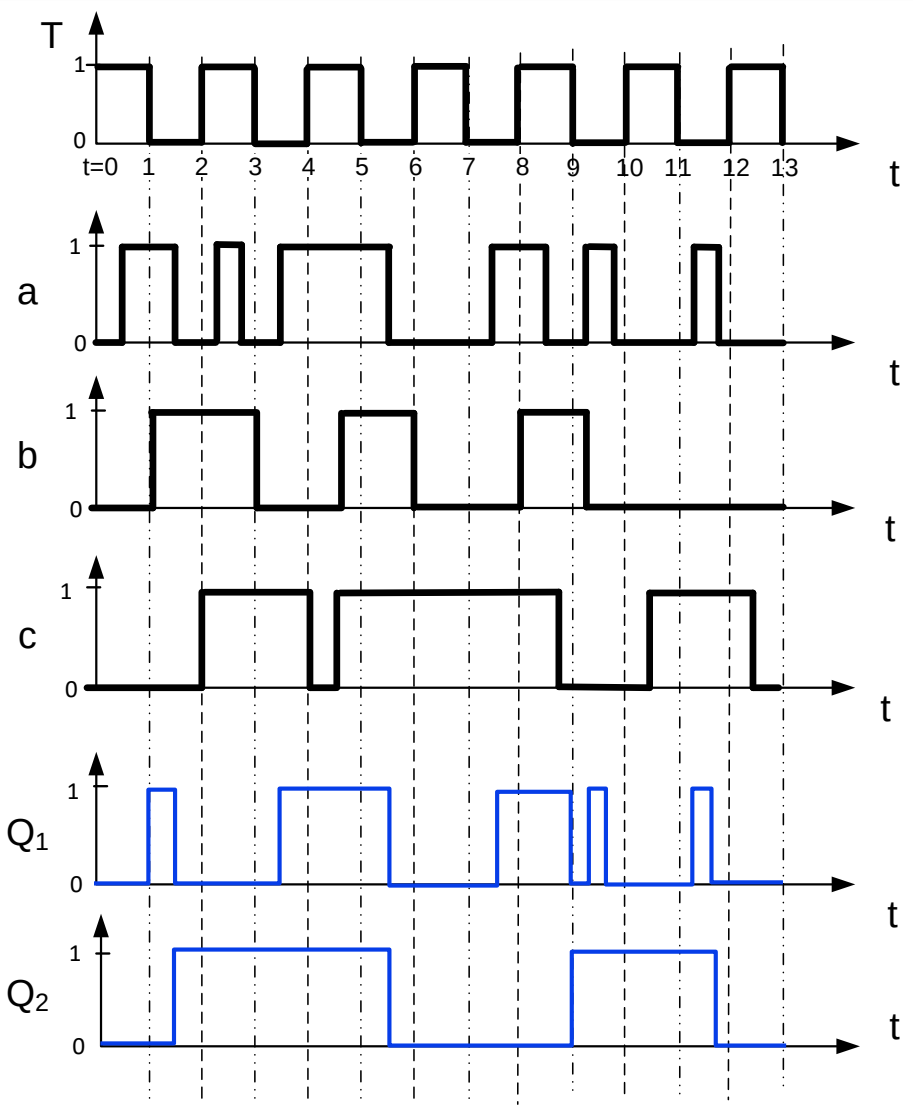


Bild G-7.1: *MS-FF*

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

Analysieren Sie die Schaltung für den Bereich $t = [0; 13[$, indem Sie für die Eingangssignale a, b, c und T die zeitlichen Verläufe für Q_1 und Q_2 in die vorgegebenen Koordinatensysteme eintragen.

Hinweis: Signallaufzeiten können bei der Analyse vernachlässigt werden.



8. MMIX-Rechner

Gegeben sei der nachfolgende Algorithmus sowie ein Ausschnitt der MMIX-Code-Tabelle (Bild G-8.1), eines Register- (Bild G-8.2) sowie eines Datenspeichers (Bild G-8.3):

$$\left((2a - b) * c - 256\right)^2$$

	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-8.1: *MMIX-Code-Tabelle*

Registerspeicher		
Adresse	Wert vor Befehlsausführung	Kommentar
…	…	…
$0x86	0x00 00 00 00 00 00 62 02	Nicht Veränderbar
$0x87	0x00 00 00 00 00 00 00 06	Variable a
$0x88	0x00 00 00 00 00 00 00 0B	Variable b
$0x89	0x00 00 00 00 00 AA 01 01	Variable c
$0x8A	0x00 00 00 00 00 00 61 FE	Zwischenergebnis
$0x8B	0x00 00 00 00 00 00 01 00	Nicht Veränderbar
…	…	

Bild G-8.2: *Registerspeicher*

a) Übersetzen Sie den gegebenen Algorithmus in MMIX-Code mit maximal 5 Anweisungen. Verwenden Sie dazu lediglich die in Bild G-8.1 markierten Befehle. Im Registerspeicher eines MMIX-Rechners befinden sich die in Bild G-8.2 gegebenen Werte. In der Spalte *Kommentar* wurde angegeben, welche Daten diese enthalten und wofür die einzelnen Zellen benutzt werden müssen. Speichern Sie die Zwischenergebnisse nach jedem Befehl in der Registerzelle mit dem Kommentar *Zwischenergebnis*.

```
1  MULI $0x8A $0x87  0x02
2  SUB  $0x8A $0x8A $0x88
3  MUL  $0x8A $0x8A $0x89
4  SUB  $0x8A $0x8A $0x8B
5  MUL  $0x8A $0x8A $0x8A
```

ADD $0x8A $0x87 $0x87 auch richtig

Reihenfolge vertauschbar

b) Speichern Sie den Inhalt der Registerspeicherzelle *Zwischenergebnis* nach Ausführung des Algorithmus ausgehend von den vorgegebenen Registerwerten als Okta im Datenspeicher ab Adresse 0x0 … 62 00. Wie lautet der vollständige MMIX-Befehl zum Speichern? Welcher Wert befindet sich zum Zeitpunkt des Speicherns in Registerspeicherzelle $0x8A? Welche Werte befinden sich nach Ausführung des Speicherbefehls im Datenspeicher?

Theoretisch auch bis 0x05 möglich

Befehl:

STOI $0x8A $0x86 0x00

Wert Zwischenergebnis $0x8A:

$(0x\ 00\ 00\ 00\ 00\ 00\ AA\ 00\ 01)_2$
Bzw. 0x 00 00 70 E4 01 54 00 01

Datenspeicher	
Adresse	Wert
…	…
0x00 00 00 00 00 00 61 FF	
0x00 00 00 00 00 00 62 00	**00**
0x00 00 00 00 00 00 62 01	**00**
0x00 00 00 00 00 00 62 02	**70**
0x00 00 00 00 00 00 62 03	**E4**
0x00 00 00 00 00 00 62 04	**01**
0x00 00 00 00 00 00 62 05	**54**
0x00 00 00 00 00 00 62 06	**00**
0x00 00 00 00 00 00 62 07	**01**
0x00 00 00 00 00 00 62 08	
0x00 00 00 00 00 00 62 09	
…	…

Bild G-8.3: *Datenspeicher*

Aufgabe BS: Betriebssysteme

Aufgabe BS:
51 Punkte

9. Speicherreservierung

a) Gegeben sei der folgende Speicherzustand. Vervollständigen Sie den Ablauf mittels der Methode Least Recently Used (LRU).

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

Zeit		0	1	2	3	4
Referenz		-	0	4	7	7
Seite 1		3	3	3	7	7
Seite 2		2	0	0	0	0
Seite 3		4	4	4	4	4
Seite 4		5	5	5	5	5
Stapel	0	5	0	4	7	7
	1	4	5	0	4	4
	2	3	4	5	0	0
	3	2	3	3	5	5

b) Beantworten Sie die nachfolgenden Fragen zum Thema Speicherreservierung und Adressumformung.

Welches der folgenden Adressierungselemente verweist auf einen Speicherbereich fester Größe? Welche Elemente befinden sich im realen bzw. virtuellen Speicher?

	Größe		Speicher	
	fest	**variabel**	**real**	**virtuell**
Rahmen	(X)	( )	(X)	( )
Segment	( )	(X)	( )	(X)
Seite	(X)	( )	( )	(X)

10. Asynchrones Scheduling, präemptiv, Round Robin (RR)

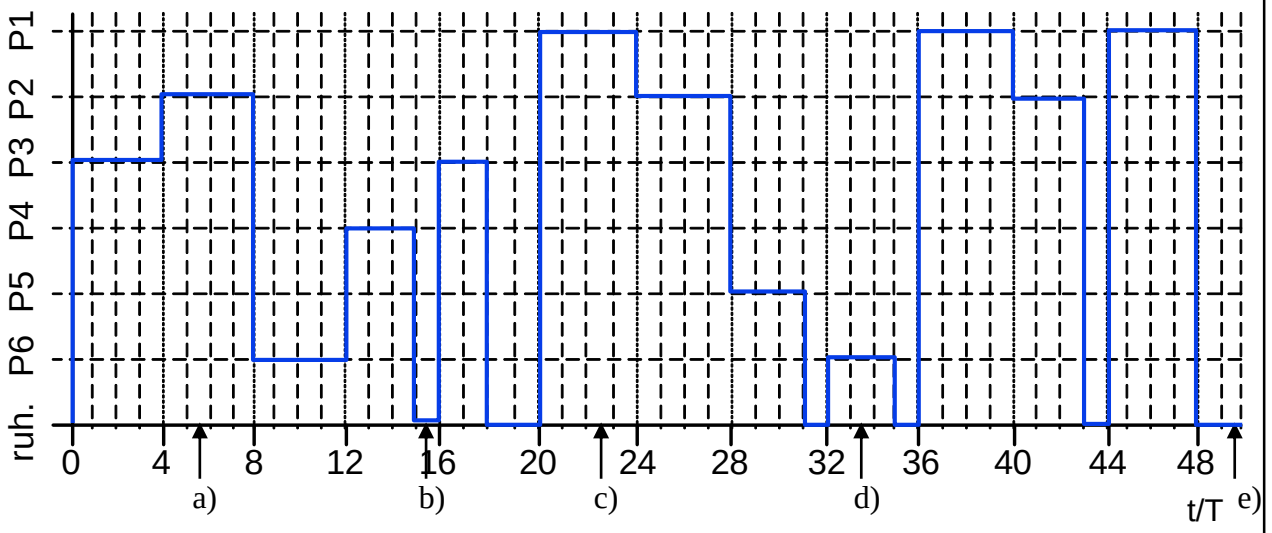
Gegeben seien die folgenden sechs Prozesse (Bild BS-10.1), welche jeweils ab dem Zeitpunkt „Start“ eingeplant werden sollen. Zur Abarbeitung eines Tasks wird die Rechenzeitspanne „Dauer“ benötigt. Periodische Tasks werden mit der Häufigkeit „Frequenz“ erneut aufgerufen. Erstellen Sie im untenstehenden 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)	7 s	4 s	17 s
P2	2	1 s	11 s	einmalig
P3	3	0 s	6 s	einmalig

	Priorität	Start	Dauer	Frequenz
P4	4	4 s	3 s	einmalig
P5	5	9 s	3 s	einmalig
P6	6 (niedrig)	3 s	7 s	einmalig

Bild BS-10.1: *Taskspezifikation*



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

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

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

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

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

11. Semaphoren

a) Gegeben seien die folgenden vier Tasks T1 bis T4 mit absteigender Priorität sowie die dazugehörigen Semaphoren S1 bis S4 (Bild BS-11.1). 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(S4)
			P(S4)
…	…	…	…
	V(S1)		V(S3)
V(S3)	V(S3)	V(S4)	V(S3)

Bild BS-11.1: *Semaphorenzuweisung*

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

b) Wie lautet der Befehl in PEARL zur Deklaration der nicht-globalen Semaphore S2 mit Startwert 1?

DCL S2 SEMA PRESET 1;

12. Echtzeitbetriebssysteme

Nachfolgend finden Sie mehrere PEARL-Anweisungen. Ordnen Sie die jeweils betroffenen Tasks deren aktuellen Zustand im erweiterten Taskzustandsdiagramm von RTOS-UH unmittelbar nach Ausführung der einzelnen PEARL-Anweisungen zu.

a) Task1 : TASK; END;

(X) ruhend ( ) bereit ( ) laufend

( ) eingeplant ( ) blockiert ( ) blockiert und eingeplant

b) REQUEST S1; *(erfolglos)*

( ) ruhend ( ) bereit ( ) laufend

( ) eingeplant (X) blockiert ( ) blockiert und eingeplant

c) TERMINATE Task1;

(X) ruhend ( ) bereit ( ) laufend

( ) eingeplant ( ) blockiert ( ) blockiert und eingeplant

d) SUSPEND Task1;

( ) ruhend ( ) bereit ( ) laufend

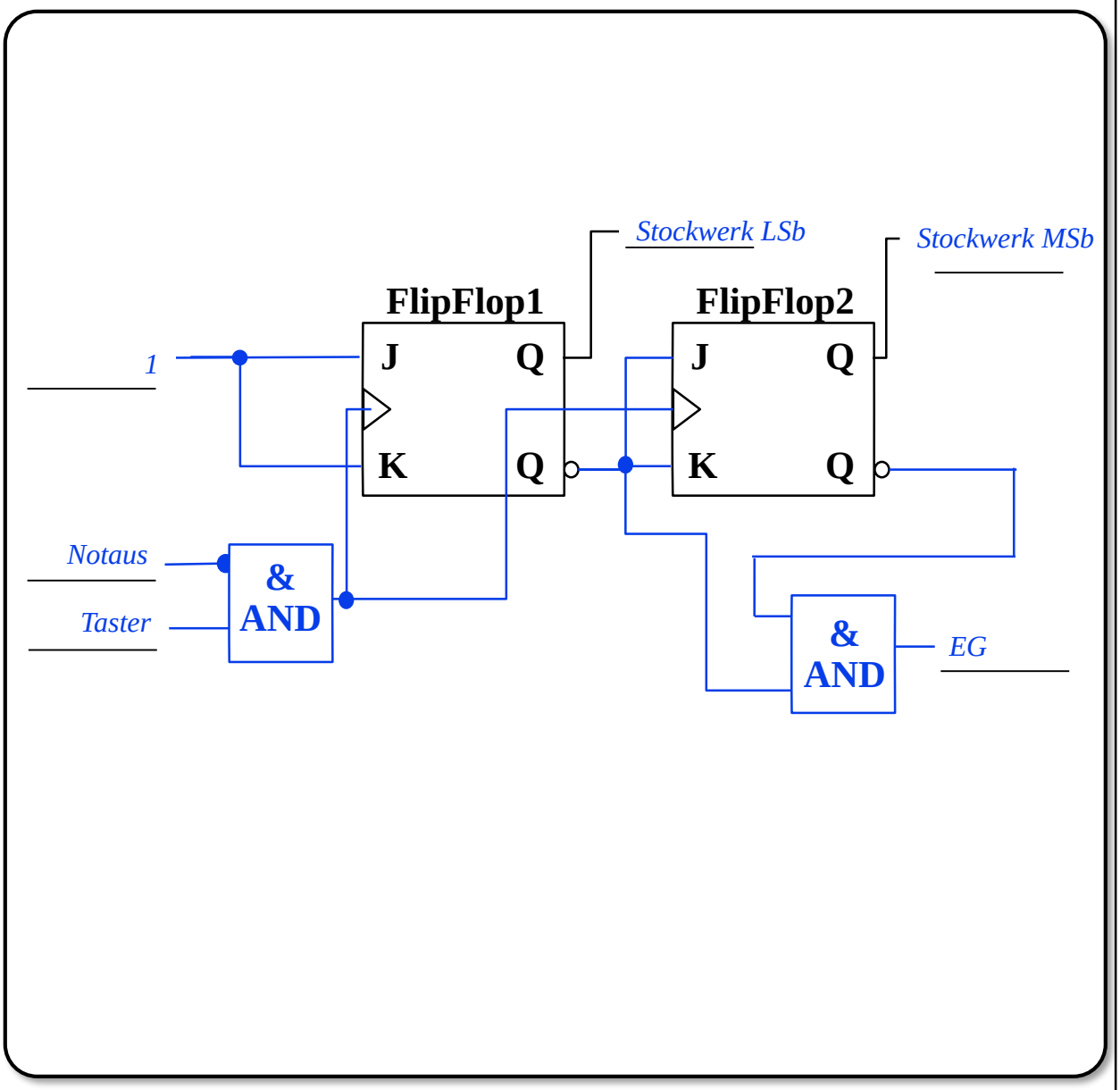
( ) eingeplant (X) blockiert ( ) blockiert und eingeplant

13. IEC 61131-3: Funktionsbausteinsprache (FBS)

Die Steuerung eines Aufzugs soll mittels FBS der IEC61131-3 implementiert werden. Der Aufzugantrieb soll 4 Etagen anfahren können. Das Sollstockwerk wird dem Antrieb binär mittels der zwei Signale *Stockwerk LSb* (least significant bit) und *Stockwerk MSb* (most significant bit) übermittelt. Bei Druck auf die *Taste* soll sich der Aufzug ins nächst niedrigere Stockwerk bewegen. Ist der Aufzug im untersten Stockwerk angekommen, bewegt sich dieser beim nächsten Tastendruck wieder in die oberste Etage. Die Abfolge der Stockwerke ist somit 11, 10, 01, 00, 11…

Befindet sich der Fahrstuhl im Stockwerk 00 soll zudem das Signal *EG* wahr sein. Die *Taste* darf nur dann auf einen Tastendruck reagieren, wenn das Signal *Notaus* gleichzeitig falsch ist.

Hinweis: Signalverzögerungen im System sind zu vernachlässigen, zulässige Signalnamen sind *Notaus*, *Taste*, *1*, *Stockwerk LSb*, *Stockwerk MSb* und *EG*



Aufgabe MSE: Modellierung und Softwareentwicklung

Aufgabe MSE:
44 Punkte

14. Automaten

a) Handelt es sich bei dem in Bild MSE-14.1 gegebenen Automaten um einen Mealy- oder einen Moore-Automaten?

Moore	X
Mealy	

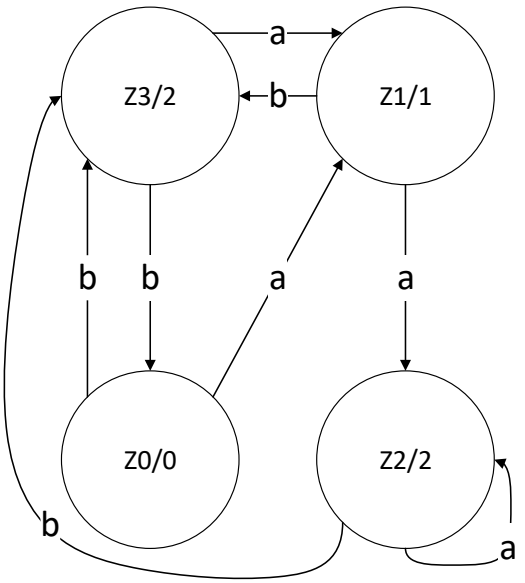
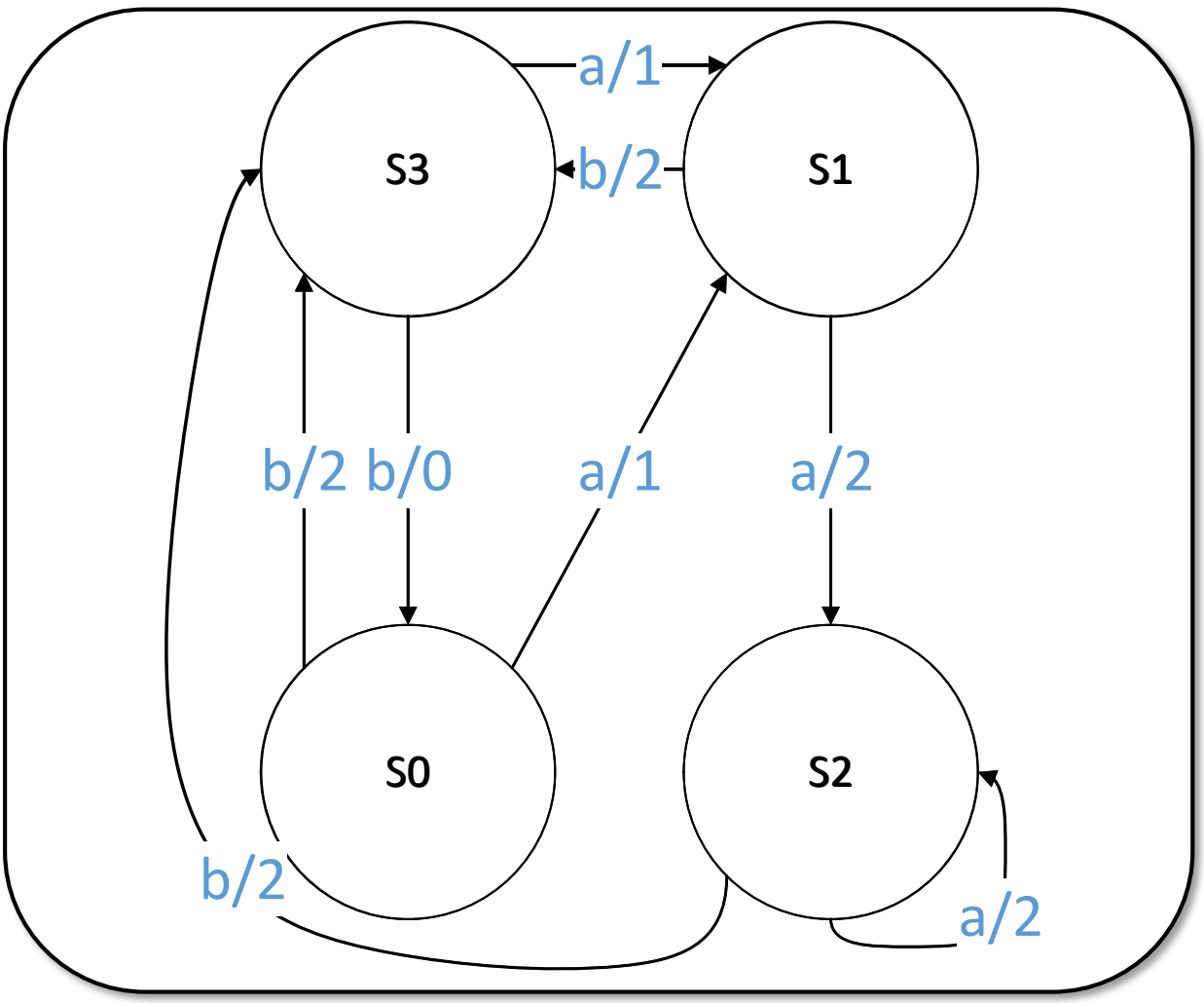


Bild MSE-14.1: *Automat*

b) Führen Sie den gegebenen Automaten in die andere Darstellung über. Beschriften Sie dazu die fehlenden Transitionen.



c) Welche beiden Zustände des vorherigen Automaten können Sie durch Zusammenfassung vereinfachen?

S0 ( )
S1 (X)
S2 (X)
S3 ( )

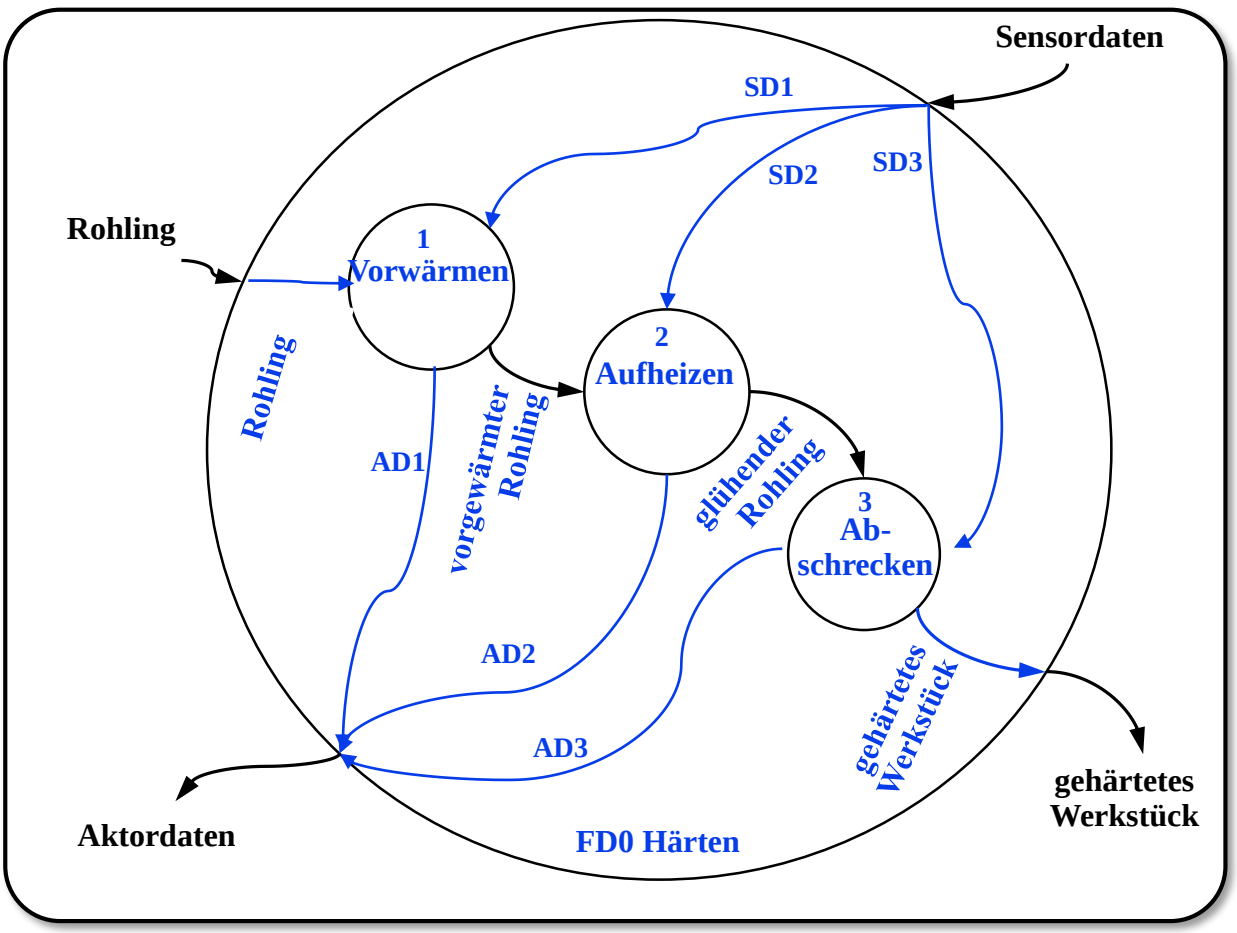
15. SA/RT: Flussdiagramm

Für die folgenden Teilaufgaben soll ein Prozess zur Härtung metallischer Werkstücke betrachtet werden, der aus drei Prozessschritten besteht.

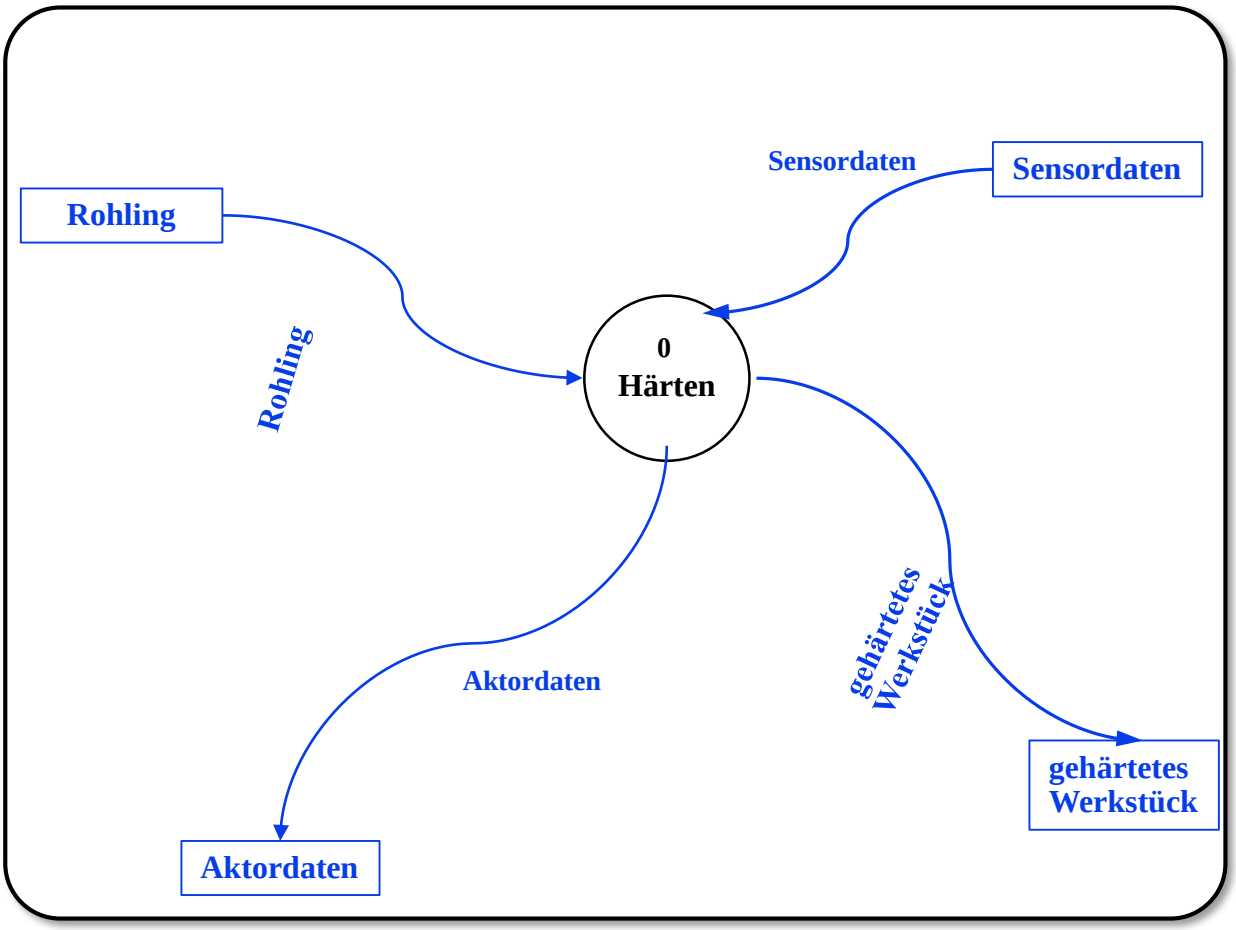
Im Prozess *Vorwärmen* (Prozess 1) wird ein *Rohling* kontrolliert *vorgewärmt*. Im Prozess *Aufheizen* (Prozess 2) wird der *vorgewärmte Rohling* hoch erhitzt und zum *glühen* gebracht. Im Prozess *Abschrecken* (Prozess 3) wird schließlich der *glühende Rohling* durch *Abschrecken* zum *gehärteten Werkstück* verarbeitet.

a) Modellieren Sie den Prozess *Härten* (Prozess 0) mittels Strukturierter Analyse / Real-Time (SA/RT) in einem Flussdiagramm (FD0). Identifizieren Sie hierzu alle Subprozesse sowie Datenflüsse 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 T3) zusammengefasst werden. Beachten Sie außerdem folgende Informationen:

Flüsse Material: Rohling, vorgewärmter Rohling, glühender Rohling, gehärtetes Werkstück	**Flüsse Sensordaten:** je nach Prozess: SD*Prozessnummer*, also z.B. SD2 bei „Vorwärmen“. **Flüsse Aktordaten:** Wie Sensordaten nur AD*Prozessnummer*



b) Erstellen Sie für den zuvor beschriebenen Prozess *Härten* das zugehörige Kontextdiagram.



16. SA/RT: Antwortzeitspezifikation, Timing-Diagramm

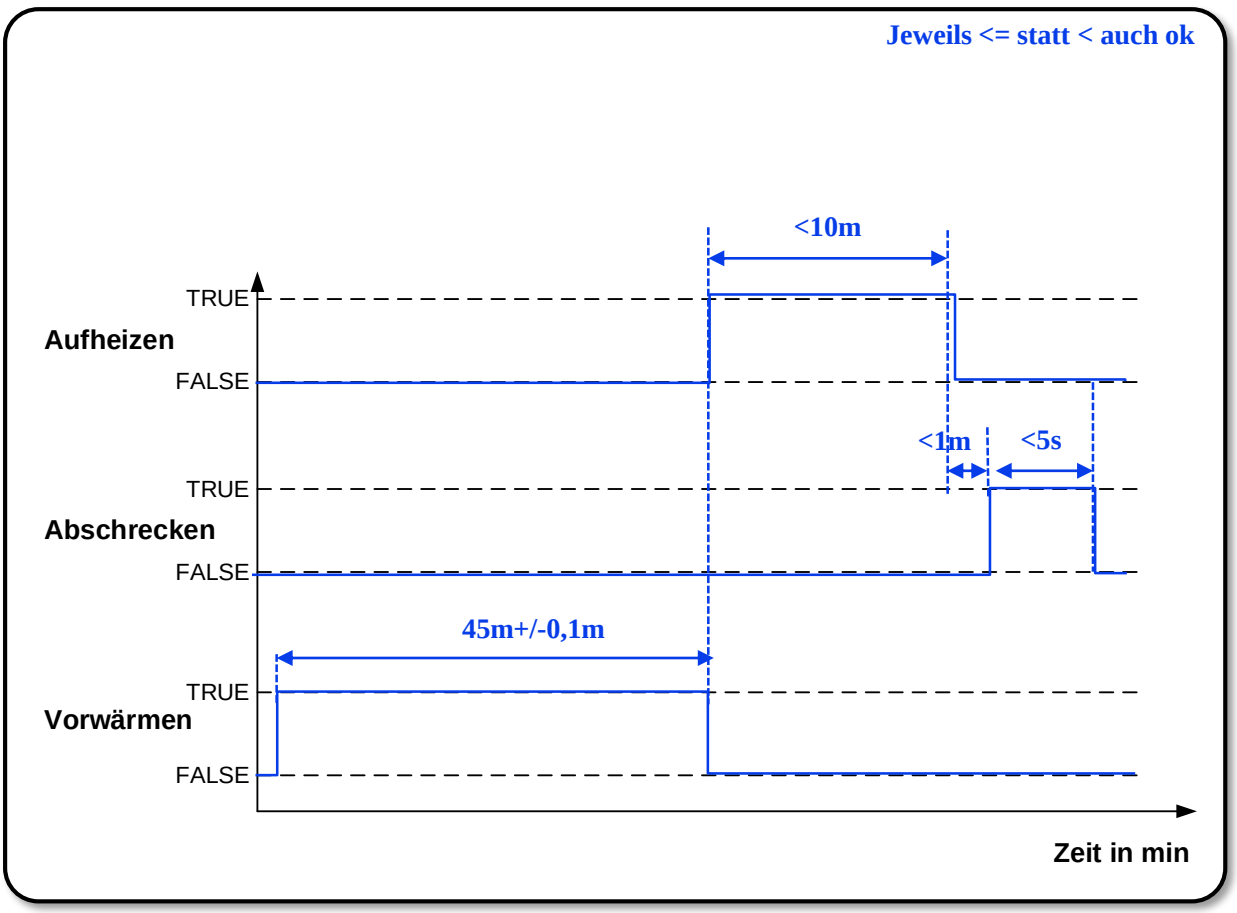
Nachfolgend soll der zeitliche Ablauf des Härtungsvorgangs in Form eines Timing-Diagramms erstellt werden, um sicherzustellen, dass der Prozess korrekt durchgeführt wird.

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

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

- Sobald das Vorwärmen begonnen wurde (*Vorwärmen=TRUE*), muss nach 45 Minuten mit einer Toleranz von 0,1 Minuten das *Vorwärmen* beendet und das *Aufheizen* begonnen werden. Das *Aufheizen* ist kürzer als 10 Minuten.
- Das *Abschrecken* beginnt spätestens eine Minute nach Ende des *Aufheizens* und darf höchstens 5 Sekunden andauern.

Hinweis: Eine maßstabsgetreue Darstellung der Zeiten ist nicht notwendig.



17. Bussysteme

Ordnen Sie die nachfolgenden Aufgaben den zuständigen Schichten des ISO 7498-Schichtenmodells zu

Hinweis: Nur Antworten innerhalb der Lösungskästen werden gewertet!

	Applikationsschicht	Darstellungsschicht	Sitzungsschicht	Transportschicht	Netzschicht	Sicherungsschicht	Bitübertragungsschicht
Konvertierung von Big und Little Endian	( )	(x)	( )	( )	( )	( )	( )
Bereitstellung von Anwendungsdiensten	(x)	( )	( )	( )	( )	( )	( )
Routing / Relaying von Datenpaketen durch Netze, via Transitknoten	( )	( )	( )	( )	(x)	( )	( )

Der Profibus-DP setzt u.a. Token Passing als Buszugriffsverfahren ein **wahr (x)** **( ) falsch**

18. Unified Modeling Language

a) Das objektorientierte Paradigma lässt sich anhand von sieben Grundkonzepten beschreiben. Nennen Sie drei dieser Grundkonzepte.

Relation, Verfeinerung, Vererbung, Abstraktion, Polymorphie, Klassenbildung, Kapselung

b) Nennen Sie je zwei Beispiele für Verhaltensdiagramme und Strukturdiagramme in der UML

Verhaltensdiagramme:

- **Zustandsdiagramm**
- **Aktivitätsdiagramm**

Strukturdiagramme:

- **Klassendiagramm**
- **Kompositionsstrukturdiagramm**

Aufgabe C: C-Programmierung

Aufgabe 19:
11 Punkte

19. C: Datentypen, Operatoren und Boolsche Algebra

a) Datentypen und Operatoren
Welche Ausgaben erzeugen die `printf`-Anweisungen in den folgenden Codefragmenten? Wählen Sie die korrekte Antwort (nur Einfachantwort möglich), bzw. füllen Sie die Lücken.

```
int x = 8, y = 6;
float f = x / y;
printf("%.0f", f);
```

( ) 0
(x) 1
( ) 1.000000
( ) 1.333333

```
int x = 2, y = 2;
int *c = &x;
float z = 3.566;
printf("%i", *c + y);
```
→ **Ausgabe:** 4

```
printf("%.2f", *c + z);
```
→ **Ausgabe:** 5.57

Sortieren Sie die Datentypen **aufsteigend** nach Wertebereich. Füllen Sie hierfür Zahlen von 1 (kleinster Wertebereich) bis 5 (größter Wertebereich) in die Lücken hinter den Datentypen ein.

char (1) double (5) short (2) float (4) int (3)

b) Ein- und Ausgabe
Ergänzen Sie das folgende Codefragment, so dass `3.2` ausgegeben wird. Wählen Sie hierfür die korrekten Alternativen in Abbildung C-19.1 aus (nur Einfachantwort möglich).

```
float x = 3.246;
_____(1)_____("_____(2)_____", x);
```

1
( ) fprintf
( ) print
( ) scanf
(x) printf

2
( ) %1.f
( ) %f.1
(x) %.1f
( ) %1.2f

Abbildung C-19.1: Antwortalternativen (nur Einfachantwort möglich).

c) Boolsche Algebra
Bestimmen Sie das Ergebnis der folgenden Ausdrücke im Dezimalsystem. Gegeben sind folgende Variablen:

```
int a = 6;
int b = 2;
int c = 1;
int *d = &b;
```

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

c.1	`a & b | c`	**3**
c.2	`(a >> *d) == c) >= ((b >> 1) + (c << 1))`	**0**

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 doppelt so groß wie iZahl2 und größer als iZahl3

```
(iZahl1 == iZahl2 * 2) && iZahl1 > iZahl3
```

d.2) iZahl2 ist ungerade oder iZahl1 ist gerade

```
(iZahl2 % 2 != 0) || (iZahl1 % 2 == 0)
```

Aufgabe 20:
12 Punkte

20. Kontrollstrukturen

Sie arbeiten für eine Versicherung, die Hausrat gegen Gefahren durch Gewitter und Überschwemmungen versichert. Sie wurden beauftragt, eine Software zu erstellen, die die Daten einer Wetterstation entgegennimmt und abhängig vom Vorhandensein bestimmter Wetterphänomene die Versicherungsprämie berechnet.

Bitte wählen Sie die korrekte Antwortmöglichkeit (nur Einfachantwort möglich) oder füllen Sie die Lücken in den Teilaufgaben.

Sie erstellen das Gesamtprogramm in drei Abschnitten (Teilaufgaben a), b) und c)). Beachten Sie, dass Code, der in einer früheren Teilaufgabe gegeben oder erstellt wurde, auch in den folgenden Teilaufgaben gilt.

a) Sie erstellen zunächst das Codegerüst. Binden Sie benötigte Bibliotheken ein, deklarieren Sie die `main`-Funktion und beenden Sie das Programm mit dem Rückgabewert für erfolgreiches Beenden.

```
     (1)
_____________
#define TAGE 4

int main() {
    printf(“Praemienrechner 1.0.\n");
    //Wird in Teilaufgabe b) implementiert.
    //Wird in Teilaufgabe c) implementiert.

    ____(2)____;
}
```

Lücke Nr. 1

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

Lücke Nr. 2: **return 0**

Abbildung C-20.1: Codefragment für Aufgabe 20 a)

b) Schreiben Sie eine `for`-Schleife, die alle aufgezeichneten Wetterdaten durchläuft. Initialisieren Sie hierfür das Array `wetterDaten` mit den aufgezeichneten Daten (siehe Abbildung C-20.2, Initialisierung in Lücke 3). Bei einem Niederschlag von mehr als 60mm und mehr als 2 Blitzen pro Tag wird ein Gewitter vermerkt. Bei mehr als 3 erkannten Gewittern, wird die Schleife verlassen, die Versicherungsprämie auf 400€ gesetzt und das Programm fortgesetzt. Bitte beachten Sie, dass der Code aus Aufgabe a) weiter gilt.

	Niederschlag	Blitze
1	66	3
2	2	2
3	6	3
4	62	5

***Abbildung C-20.2*:** Wetterdaten.

```
//[Tag][0] -> Niederschlag, [Tag][1] -> Anzahl Blitze
int wetterDaten[TAGE][2] =     _____(3)_____;
int niederschlag = 0, gewitter = 0, blitze = 0, praemie = 0;

for (_____(4)_____) {
        niederschlag += wetterDaten[i][0];
        blitze += wetterDaten[i][1];

        if (wetterDaten[i][0] > 60 && wetterDaten[i][1] > 2) {
            gewitter++;
            printf("Gewitter an Tag %i mit %i Blitzen.\n",
                        _____(5)_____);
        }
        if (gewitter > 3) {
            printf("Hohe Gewittergefahr.");
            ____(6)____;
            ____(7)____;
        }
}
```

Lücke Nr. 3: `{{66, 3}, {2, 2}, {6, 3}, {62, 5}}`

Lücke Nr. 4

```
( ) i = 1; i < TAGE; i+1        ( ) i = 1; i < TAGE; i++
( ) i = 0; i > 4; i--           (x) i = 0; i < TAGE; i++
( ) i = 0; i < 2; i++           ( ) i = 0; i <= TAGE; i++
```

Lücke Nr. 5

```
( ) i, wetterDaten[i + 1][0]    ( ) i + 1, wetterDaten[i + 1][1]
(x) i + 1, wetterDaten[i][1]    ( ) i + 1, wetterDaten[i + 1][0]
```

Lücke Nr. 6: `praemie = 400`

Lücke Nr. 7: `break`

Abbildung C-20.3: Codefragment für Aufgabe 20 b)

c) Nun soll die Versicherungsprämie in Abhängigkeit der Wetterdaten am Standort des Hauses berechnet werden. Wenn pro Tag durchschnittlich mehr als 3 Blitze und 1 Gewitter registriert wurden, wird die Prämie um 100€ erhöht. Bei einem Niederschlag von mindestens 200mm werden 50€ aufgeschlagen.

Bitte beachten Sie dabei die Datentypen und Startwerte der in Abbildung C-20.1 und C.-20.3 deklarierten Variablen.

```
if (_____(8)_____ > 3 && gewitter > 1) {
    preemie += 100;
}
if (niederschlag (9)_____) {
    praemie += 50;
}
printf("Die Praemie betraegt: (10)___ Euro", praemie);
```

Lücke Nr. 8: (float) blitze / TAGE

Lücke Nr. 9: >= 200

Lücke Nr. 10:

( ) %f	( ) %c	( ) &f
( ) %lf	(x) %i	( ) &d

Abbildung C-20.4: Codefragment für Aufgabe 20 c)

21. Objektorientierte Programmierung

Aufgabe 21:
23 Punkte

a) Grundlagen & Konzepte

Folgend werden grundlegende Konzepte der objektorientierten Programmierung abgefragt. Bitte wählen Sie aus den Antwortalternativen die korrekte Alternative aus (nur Einfachnennung möglich), oder füllen Sie die markierten Lücken mit dem korrekten Begriff. Aufgaben 21a) 1 - 3 beziehen sich auf das in Abbildung C-21.1 abgebildete Klassendiagramm.

1. Wählen Sie die falsche Aussage zur Beziehung zwischen *Telefonbuch* und *Buch*.
( ) *Telefonbuch* erbt von *Buch*.
( ) *Buch* ist abstrakter als *Telefonbuch*.
(x) Jedem *Buch* ist genau ein *Telefonbuch* zugeordnet.
( ) *Telefonbuch* kann Methoden von Buch überschreiben.
( ) Jedes *Telefonbuch* ist auch ein *Buch*.
( ) *Telefonbuch* erbt die Attribute von *Buch*.

2. Die Beziehung zwischen *Telefonnummer* und *Telefonbuch* ist eine Aggregation

3. Wer kann auf die privaten Attribute von *Telefonbuch* zugreifen?
( ) Instanzen von *Buch*
(x) Instanzen von *Telefonbuch*
( ) Instanzen von *Telefonnummer*
( ) Instanzen jeder Klasse
( ) Niemand
( ) Instanzen aller anderen Klassen

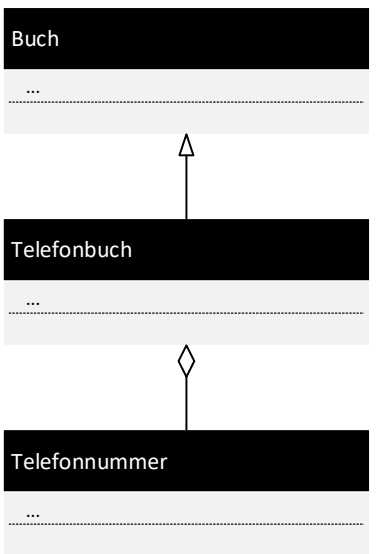


Abbildung C-21.1:
UML-Klassendiagramm.

4. Wann wird ein Konstruktor aufgerufen?
( ) Bei der Zerstörung der Instanz
( ) Abhängig vom System
(x) Bei der Instanziierung der Instanz
( ) Je nach Implementierung
( ) Bei erstmaliger Verwendung der Instanz
( ) Bei jeder Verwendung der Instanz

5. 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.

b) Modellierung mit UML

Sie sollen eine Software zum Umgang mit Wetterdaten entwickeln. Zunächst erfragen Sie von einem Experten die abzubildenden Konzepte. Ein Ausschnitt der Interviewergebnisse, die Sie in ein Klassendiagramm überführen sollen, ist in Abbildung C-21.2 gegeben.

- Das System muss Wetterphänomene speichern. Für die erste Version der Software müssen nur Stürme und Gewitter abgebildet werden.
- Jedes Phänomen ist einer Wetterstation zugeordnet, die von einem Meteorologen betreut wird.
- Eine Wetterstation wird an einem bestimmten Ort aufgestellt und verfügt über eine Funktion, die alle gespeicherten Wetterdaten (vom Typ `Wetter`) zurückgibt.

Abbildung C-21.2: Interviewergebnisse, die in ein Klassendiagramm überführt werden.

Füllen Sie die mit Zahlen markierten Lücken im folgenden Klassendiagramm (Abbildung C-21.3). Lücken **zwischen** Klassen erfordern das Einzeichnen von Beziehungen, Lücken **innerhalb** von Klassen erfordern das Einfüllen von Attributen, Methoden oder Klassennamen. Attribut- und Methodennamen sind **in dieser Teilaufgabe** nicht relevant.

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

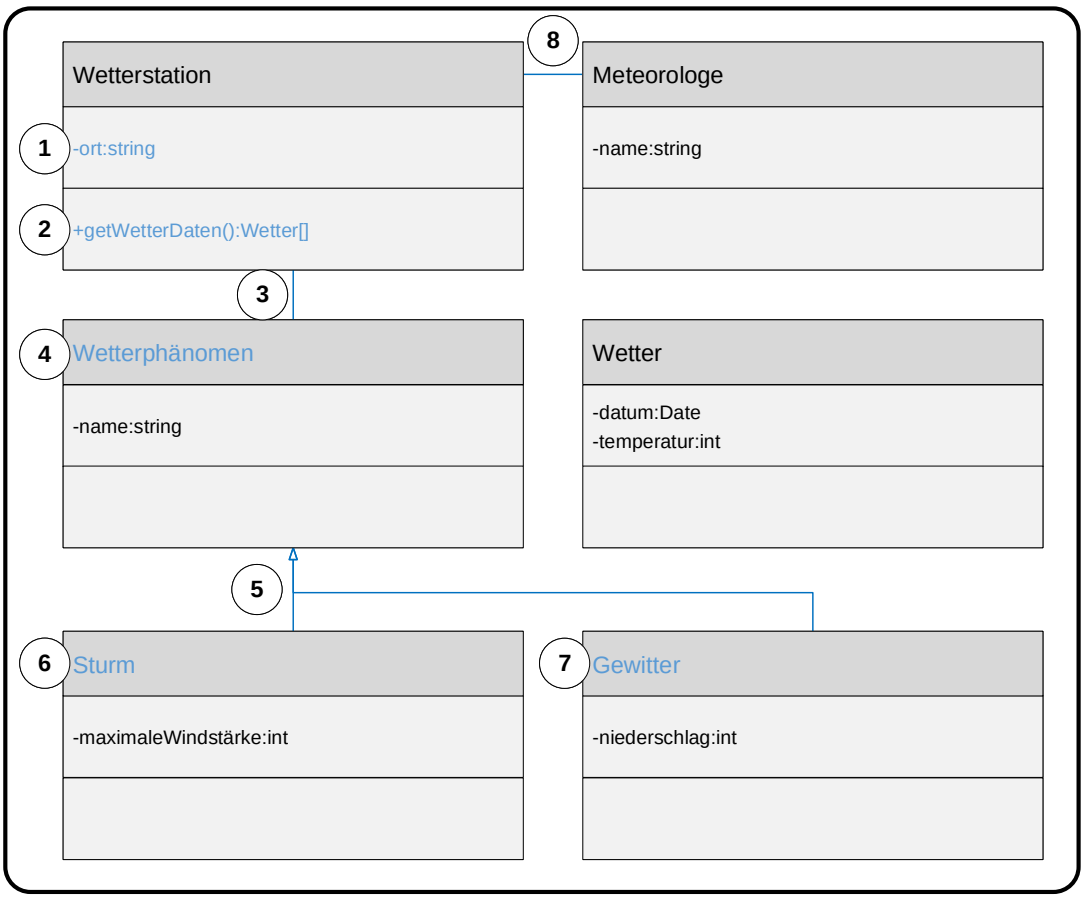


Abbildung C-21.3: Klassendiagramm der Wettersoftware.

c) Vom Code zum Klassendiagramm

Sie haben Programmcode der Wetterverwaltungssoftware erhalten. Um die Struktur der Software zu verstehen, möchten Sie ein Klassendiagramm des Codeausschnitts in Abbildung C-21.4 erstellen.

```
class Messgeraet
{
  public:
    Messgeraet(Wetterstation* station)
  private:
    string sName;
    Wetterstation* wetterstation;
};

class Niederschlagsmesser:public Messgeraet
{
  public:
    int getNiederschlag(Date datum);
  private:
};

class Wetterstation
{
  public:
    Wetterstation();
  private:
    string ort;
};
```

Abbildung C-21.4: Programmcode zur Wettersoftware.

Ein Grundgerüst des Klassendiagramms ist in Abbildung C-21.5 gegeben. 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 und Parameter, auf die Rückgabewerte der Funktionen, auf die Sichtbarkeiten von Variablen und Methoden sowie die in Teilaufgabe **b)** genannten Konventionen.

Abbildung C-21.5: Klassendiagramm für die Wettersoftware (siehe Abbildung C-21.4 auf der vorherigen Seite).

Aufgabe 22: Anlagen/Zustandsautomat

Aufgabe 22:
24 Punkte

Neben der Software, welche die Daten der Wetterstation entgegen nimmt, sollen Sie auch Teile der Wetterstation entwickeln. Eine relevante Messgröße ist hierbei die Niederschlagsmenge. Aus diesem Grund soll ein Messgerät zur Erfassung der Niederschlagsmenge automatisiert werden (siehe Abbildung C-22.1). Die Anlage sammelt den gefallenen Niederschlag in einem Tank mit definiertem Volumen und kann diesen wahlweise zunächst aufheizen und an eine nachgelagerte Analyse weiterleiten (zur Analyse) oder ableiten (zum Ablauf). Es soll die Anzahl der Tankfüllungen gezählt werden, um die Niederschlagsmenge zu erfassen.

Folgende Ein- und Ausgänge stehen Ihnen zur Verfügung:

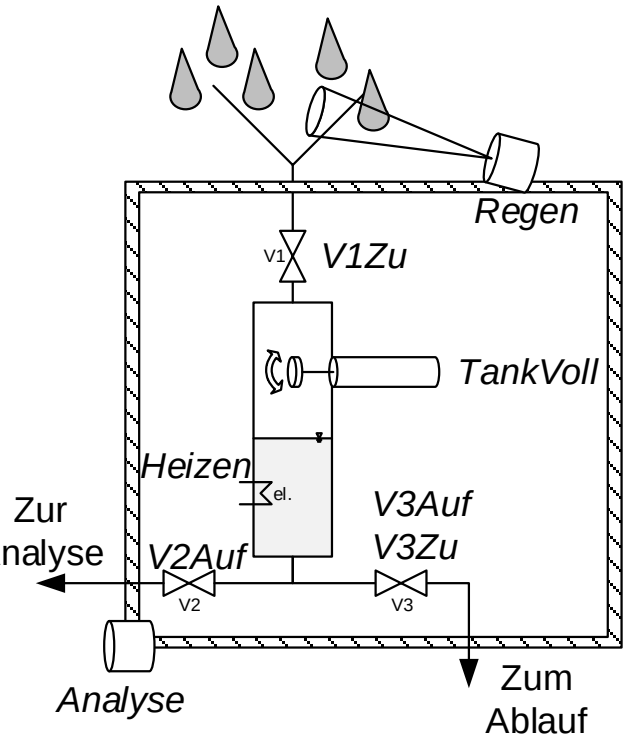


Abbildung C-22.1: Niederschlagsmessgerät

Typ	Name	Beschreibung
AKTOREN	a.V1Zu	Schließt (1) oder öffnet (0) Ventil V1
	a.V2Auf	Öffnet (1) oder schließt (0) Ventil V2
	a.V3Auf	Öffnet Ventil V3 (1) oder stoppt (0)
	a.V3Zu	Schließt Ventil V3 (1) oder stoppt (0)
	a.Heizen	Heizung an- (1) oder ausschalten (0)
SENSOREN	s.TankVoll	Liefert (1) falls der maximale Füllstand erreicht ist, sonst (0)
	s.Analyse	Anforderung der Analyse nach neuer Probe (1), sonst (0)
	s.Regen	Niederschlag wurde detektiert (1), sonst (0)
VARIABLEN	zeit	Aktuelle Laufzeit des Programms in ms
	t	Variable für timer-Programmierung
	schritt	Aktueller Zustand, welcher ausgeführt wird
	probenzahl	Zähler für erfasste Niederschlagsmessungen

Abbildung C-22.2: Sensor- und Aktorvariablen der Anlage, sowie erweiterte Variablen

Zunächst soll die Anlage in einen Initialzustand gebracht werden. Hierfür werden alle Ventile (V1–V3) geschlossen und die Heizung deaktiviert. Beachten Sie, dass die Ventile unterschiedlich angesteuert werden und V3 ein doppeltwirkendes Ventil ohne automatische Rückstellung ist.

Sobald der Sensor `Regen` Niederschlag detektiert, beginnt die Messung und V1 wird geöffnet. Es wird so lange gewartet, bis der Schwimmer `TankVoll` das Erreichen des maximalen Füllstands anzeigt und anschließend Ventil V1 geschlossen. Liegt zu diesem Zeitpunkt die Anfrage nach einer neuen Probe durch die Analyse vor (Variable `Analyse`), muss der Tank zunächst durch Aktivieren der Heizung aufgeheizt werden. Nach 30 Sekunden kann dann Ventil V2 geöffnet werden. Nach weiteren 20 Sekunden ist das Wasser abgelaufen und V2 kann wieder geschlossen werden. Weiterhin soll die Heizung erst abgeschaltet werden, nachdem der Tank entleert wurde. Die Anlage geht zurück in den Initialzustand und wartet erneut auf Niederschlag.

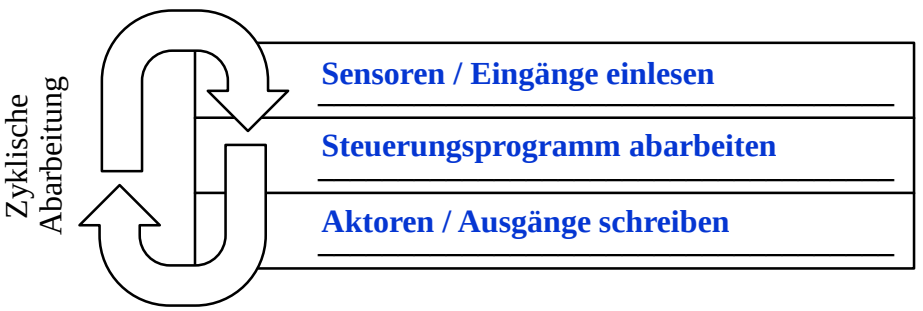
Liegt keine Anfrage der Analyse vor, soll das Wasser zum Ablauf geleitet werden. Hierzu muss Ventil V3 geöffnet werden. Nach 20 Sekunden wird dieses wieder geschlossen und die Anlage wartet im Initialzustand auf Niederschlag.

Bitte beachten Sie, dass für jede genommene Messung, egal ob diese zur Analyse oder zum Abfluss abgeleitet wird, die Variable `probenzahl` inkrementiert werden muss, um eine Statistik über den gefallenen Niederschlag zu erhalten.

Für die Programmierung einer zeitlichen Verzögerung soll die Hilfsvariable `t` verwendet werden. Die aktuelle Ausführungszeit des Programms ist in `time` gegeben.

a) Wissensfrage

I. Mit welchen Schritten und in welcher Reihenfolge wird allgemein ein Steuerungsprogramm abgearbeitet? Füllen Sie die fehlenden Angaben aus.



b) Vorgegeben ist das in Abbildung C-22.3 gezeigte Zustandsdiagramm mit den korrespondieren Zustandsnummern. Füllen Sie die durch römische Ziffern und graue Hinterlegung gekennzeichneten Lücken durch Ankreuzen (nur Einfachnennung möglich) bzw. Angabe der Lösung aus.

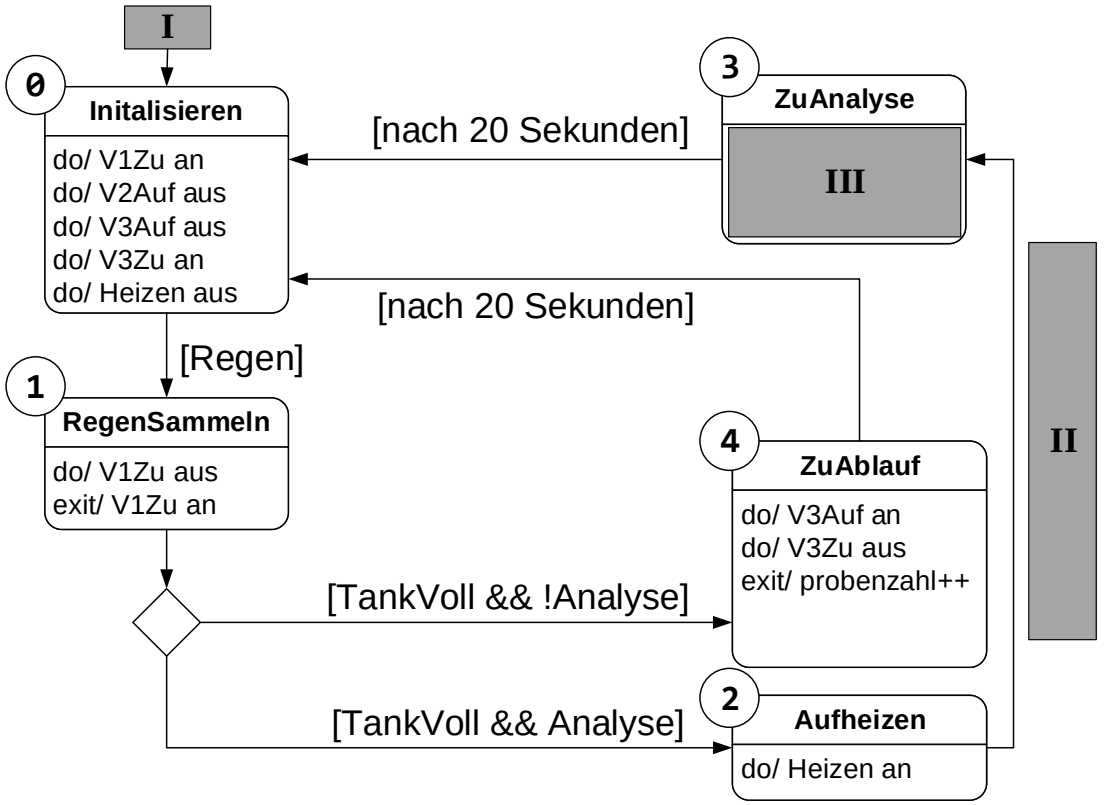


Abbildung C-22.3: Zustandsdiagramm des Niederschlagmessgeräts

I. Wählen Sie die richtige Form (nur Einfachnennung möglich).

☐ ☒ ☐ ☐

II. Geben Sie die korrekte Wächterbedingung an.

[nach 30 Sekunden]

II. Modellieren Sie den Zustand `ZuAnalyse` korrekt aus.

ZuAnalyse
do/ V2Auf an
exit/ probenzahl++

c) Programmieren Sie nun im folgenden Antwortfeld (Abbildung C-22.4) den Programmcode für den Zustand `Initialisieren`. Bitte beachten Sie, dass die gegebene Zahl an Leerzeilen nicht der Länge Ihrer Lösung entsprechen muss.

```
case 0 : //Initalisieren
   a.V1Zu = 1;
   a.V2Auf = 0;
   a.V2Zu = 1;
   a.V3Zu = 1;
   a.Heizen = 0;
   if (s.Regen)
   {
        schritt = 1;
   }
   break;
```

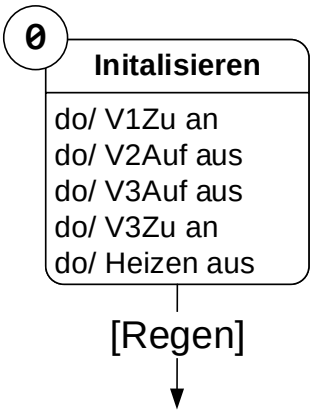


Abbildung C-22.4: Programmcode für Schritt 0 (Initialisieren).

a) Vervollständigen Sie nun den Code für den in Abbildung C-22.5 gegebenen Zustand `ZuAblauf`.

```
case 4: //ZuAblauf
      a.V3Auf = 1;
   a.V3Zu = 0;
   if      (!t)
   {
      t = zeit;
   }
   else if (zeit > t + 20000)
   {
      t = 0; probenzahl++;
      a.V3Auf = 0; a.V3ZU = 1;
      schritt = 0;
   }
   break;
```

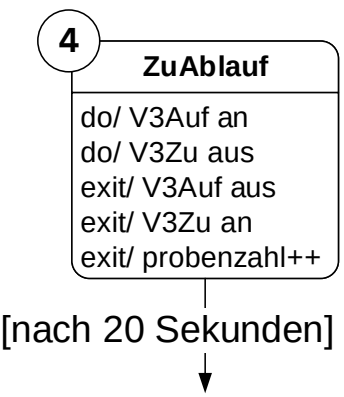


Abbildung C-22.5: Programmcode für Schritt 4 (ZuAblauf).

23. Erweiterte Datenstrukturen und FileIO

Aufgabe 23: 26 Punkte

In dieser Aufgabe schreiben Sie ein Programm, mit dem Messwerte aus einer Wetterstation eingelesen, verarbeitet und wieder in eine Datei zur weiteren Verarbeitung geschrieben werden.

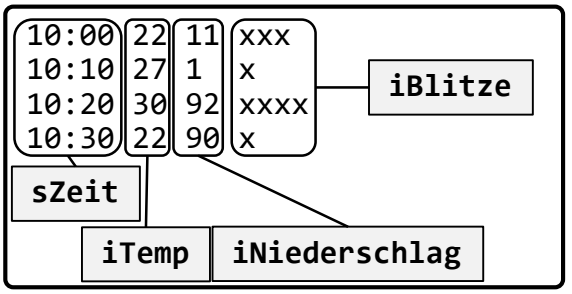


***Abbildung C-23.1*:** Auszug aus der Datei `wetterstation1.csv`.

a) Die aus der csv-Datei (siehe Abbildung C-23.1) gelesenen Messwerte sollen in einer Datenstruktur `WETTERWERT` gespeichert werden (die Werte sind durch Leerzeichen getrennt). Die Datei enthält einen Zeitstempel (`sZeit`, Zeichenkette, maximal 5 Zeichen), den gemessenen Temperaturwert (`iTemp`, Ganzzahlig), den gemessenen Niederschlag (`iNiederschlag`, Ganzzahlig, positiv) sowie die gemessenen Blitze als „Strichliste“ (ein „x“ entspricht einem Blitz).

Ergänzen Sie den untenstehenden Quelltext (Abbildung C-23.2) in der Headerdatei `datentypen.h` um die notwendigen Typdefinitionen und vervollständigen Sie die Lücken. Beachten Sie, dass die Blitze in der Datenstruktur als positive ganze Zahl gespeichert werden sollen (in diese werden sie in einer späteren Teilaufgabe umgewandelt). Verwenden Sie Präprozessordirektiven, um eine mehrfache Einbindung der Headerdatei zu verhindern.

```
#ifndef DATENTYPEN_H_INCLUDED
#define DATENTYPEN_H_INCLUDED
typedef struct {
    char sZeit[6];
    int iTemp;
    unsigned int iNiederschlag;
    unsigned int iBlitze;
} WETTERWERT;
#endif
```

Abbildung C-23.2: Typdefinition in der Datei `datentypen.h`.

b) Sie sollen nun die Datei `wetterstation1.csv` öffnen und zeilenweise einlesen. Speichern Sie die eingelesenen Daten in das Array `werte`. Die Anzahl der gemessenen Blitze ist in der Datei als Strichliste kodiert (siehe Abbildung C-23.1) wobei ein „x“ einem gemessenen Blitz entspricht. Diese Strichliste muss in eine Ganzzahl umgewandelt werden, bevor sie gespeichert werden kann.

Vervollständigen Sie den in Abbildung C-23.3 gegebenen Programmcode, bzw. wählen Sie aus den gegebenen Alternativen die korrekte Antwort aus.

Hinweis: Die Länge eines C-Strings können Sie mit der Funktion `strlen` bestimmen. Diese erwartet einen String als Parameter.

```
#include <stdio.h>
#include <string.h>
#include "datentypen.h"
#define MESSWERTE 7

int main() {
    WETTERWERT werte[MESSWERTE];
    int i = 0;
    char blitzTemp[100];
    FILE* datei = fopen("wetterstation1.csv", "r");
    for (i = 0; i < MESSWERTE; i++) {
        fscanf(datei, " %s %i %i %s",
               werte[i].sZeit,
               &werte[i].iTemp,
               &werte[i].iNiederschlag,
               blitzTemp);
        werte[i].blitze = strlen(blitzTemp);
    }
    fclose(datei); //Datei schließen
```

Abbildung C-23.3: Erster Teil des Programmcodes des Auswerteprogramms.

c) Nach dem Einlesen der Daten sollen nun der Durchschnittswert aller gemessenen Temperaturen (mit 2 Nachkommastellen), sowie die höchste gemessene Temperatur (als Ganzzahl) in die Datei `output.txt` geschrieben werden.

Hierfür wird einmal über alle im Array `werte` gespeicherten Messwerte iteriert. Speichern Sie sich einen Zeiger auf den Messpunkt mit der höchsten Temperatur (Datentyp `WETTERWERT*`) im Zeiger `maxTemp`. Berechnen Sie dann die Durchschnittstemperatur und schreiben diese und die höchste Temperatur in die Datei. Beachten Sie, dass der Code aus den vorhergehenden Teilaufgaben weiterhin gilt.

```
int iSumTemp = 0;


WETTERWERT* maxTemp = NULL;


for (i = 0; i < MESSWERTE; i++) {
 iSumTemp += ____werte[i].iTemp____;
 if (____maxTemp == NULL____||____werte[i].iTemp > maxTemp->iTemp____)
 {
     maxTemp = ____&werte[i]____; //Verweis auf Messwert speichern
 }
}


float fAVGTemp = ____(float) iSumTemp / MESSWERTE____;


FILE* datei = fopen("output.txt", "__w__");
fprintf(datei, "Durchschnittstemperatur: ____%.2f\n",____fAVGTemp____);
fprintf(datei, "Hoechste Temperatur: %i\n",____maxTemp->iTemp____);
Fclose(datei); //Datei schließen
```

Abbildung C-23.4: Zweiter Teil des Programmcodes des Auswerteprogramms.