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

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Wintersemester 2014/15

27.02.2015

Bitte legen Sie Ihren Lichtbildausweis bereit.

Sie haben für die Bearbeitung der Klausur 120 Minuten Zeit.

Diese Prüfung enthält 30 nummerierte Seiten inkl. Deckblatt.
Bitte prüfen Sie die Vollständigkeit Ihres Exemplars!

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

Diesen Teil nicht ausfüllen.

Aufgabe	GL	BS	MSE	C		∑	**Note**
erreichte Punkte							
erzielbare Punkte	48	48	48	96		240	

Nachname, Vorname _____ Matrikelnummer _____

Aufgabe GL: Zahlensysteme und Logische Schaltungen

Aufgabe GL:
48 Punkte

1. Zahlensysteme

a) Überführen Sie die unten angegebenen Zahlen in die jeweils anderen Zahlensysteme. **Hinweis***:* Achten Sie genau auf die jeweils angegebene *Basis*!

1 (23)$_{10}$ = ( )$_{8}$ = ( )$_{3}$

2 (705)$_{8}$ = ( )$_{2}$

b) Rechnen Sie die unten angegebene Zahl in eine Gleitkommazahl (angelehnt an IEEE 754) um. Die Größe der Speicherbereiche für Vorzeichen (V), Mantisse (M) und Exponent (e) entnehmen Sie der unten angegebenen Speicherdarstellung, in der Sie auch Ihr Ergebnis eintragen.

$$\mathbf{z = (-0{,}45)_{10}}$$

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

V	e (4 Bit)				M (5 Bit)				

Dualzahl (ohne Vorzeichen)

Z =

Normalisierte Gleitkommazahl

N =

Exponent E; Bias B

E = **B =**

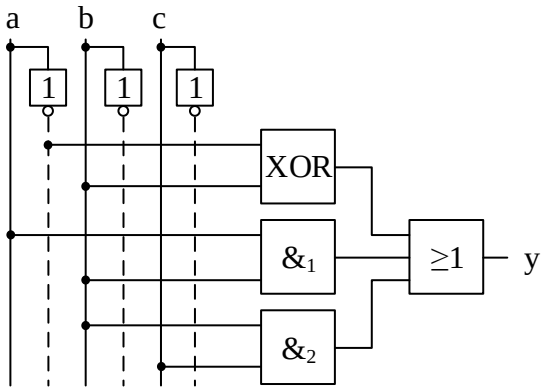
Biased Exponent b

b =

Nachname, Vorname _____ Matrikelnummer _____

2. Boolesche Algebra / Logische Schaltungen

Sie sind zuständig für die Auslegung eines Verriegelungsschutzes einer Anlage. Die elektrotechnische Abteilung übergibt Ihnen folgendes Schaltbild eines booleschen Ausdrucks.



a) Füllen Sie die Wahrheitstabelle für das Ausgangssignal y bei gegebenen Eingangsbelegungen der Variablen a, b und c aus.

Hinweis: $XOR = (\overline{x_1} \wedge x_2) \vee (x_1 \wedge \overline{x_2})$

Dez	a	b	c	XOR	&1	&2	y
0	0	0	0				
1	0	0	1				
2	0	1	0				
3	0	1	1				
4	1	0	0				
5	1	0	1				
6	1	1	0				
7	1	1	1				

Nachname, Vorname _____ Matrikelnummer _____

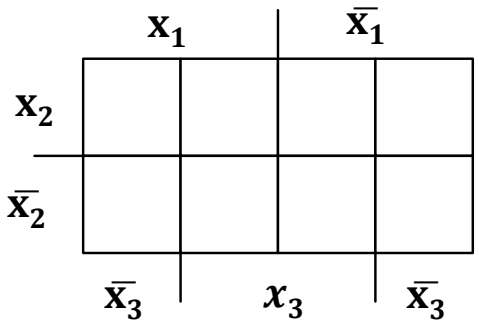
3. Normalformen und Minimierung

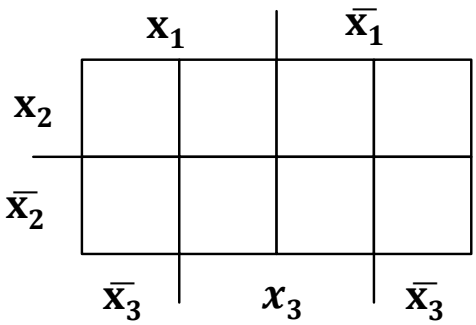
Gegeben ist die folgende Wahrheitstabelle:

Dez	x_1	x_2	x_3	y
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	X
7	1	1	1	X

a) Tragen Sie die Ausgangsvariable y aus der Wahrheitstabelle in das KV-Diagramm ein und minimieren Sie die DNF (*Disjunktive Normalform*) mit Hilfe des KV-Diagramms. Schreiben Sie ebenfalls die minimierte Funktion in boolescher Algebra auf. Die Ausgänge mit y =„X“ sind „don‘t-care“-Bits.

Hinweis: Das zweite abgebildete KV-Diagramm ist lediglich redundant, falls Sie sich verzeichnen.

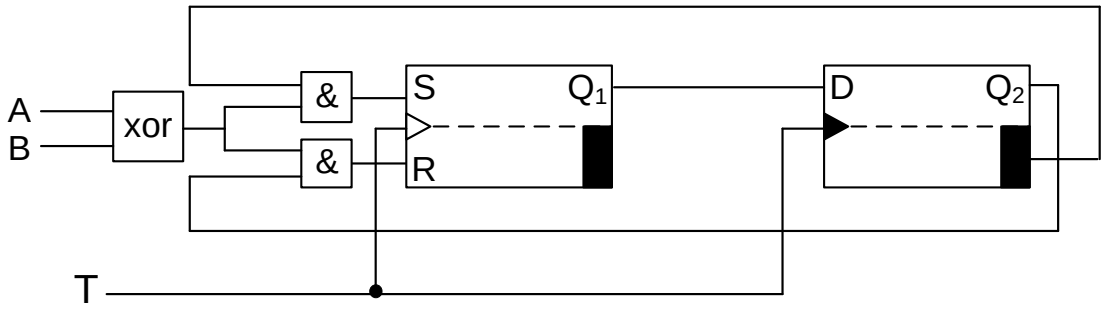




Nachname, Vorname _____ Matrikelnummer _____

4. Flip-Flops

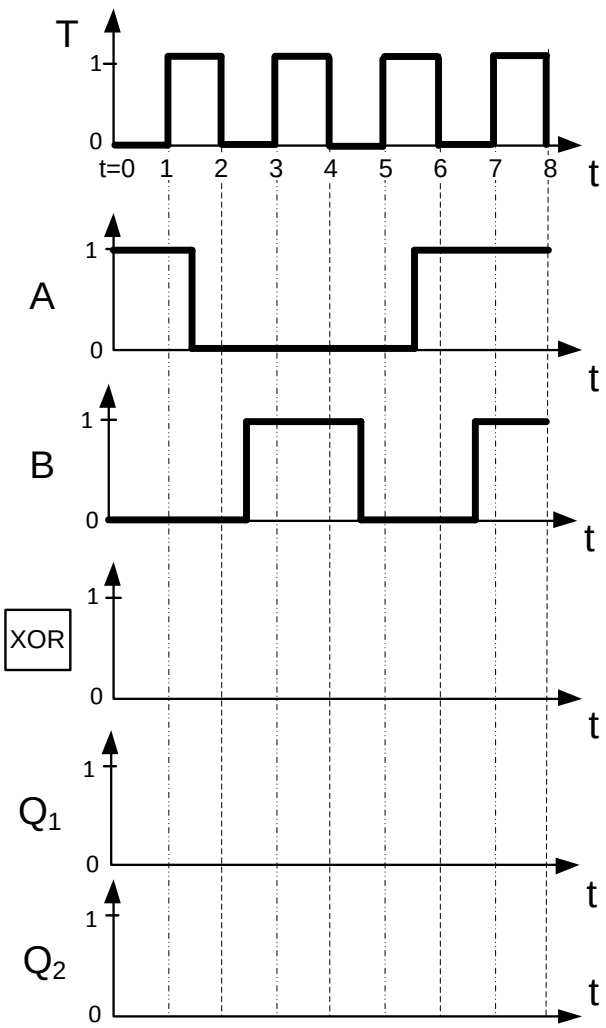
Gegeben ist die folgende Master-Slave-Flip-Flop-Schaltung.



Bei t = 0 sind die Flip-Flops in folgendem Zustand: $Q_1 = 1$ und $Q_2 = 1$.

Analysieren Sie die Schaltung, indem Sie für die Eingangssignale A, B und T die zeitlichen Verläufe für Q_1, Q_2 und XOR in die vorgegebenen Koordinatensysteme eintragen.

Hinweis: Signallaufzeiten können bei der Analyse vernachlässigt werden.



Nachname, Vorname _____ Matrikelnummer _____

5. Befehlsabarbeitung mit MMIX-Rechnerarchitektur

Zum Startzeitpunkt besitzen der Register- und der Datenspeicher eines MMIX-Rechners die in Tabelle GL-5.3 bzw. GL-5.4 (siehe folgende Seite) gegebenen Werte. Es sollen nacheinander drei Befehle ausgeführt und ein Befehl erstellt werden (Tabelle GL-5.5).

Ergänzen Sie zunächst die drei Befehle der Tabelle GL-5.5, indem Sie die gegebenen Maschinen- oder Assemblerbefehle bzw. Befehlsbeschreibung in die jeweils fehlenden Formen umwandeln (ein Beispiel finden Sie in Tabelle GL-5.2). Führen Sie dann diese Befehle mit den Werten von Register- und Datenspeicher durch (Tabelle GL-5.3 bzw. GL-5.4, „Wert vor Befehlsausführung“) und füllen Sie den Register- und den Datenspeicher für den Zustand nach der Befehlsausführung vollständig aus (Tabelle GL-5.3 bzw. GL-5.4, „Wert nach Befehlsausführung“).

Erstellen Sie abschließend einen Befehl (Maschinen- und Assemblersprache und Befehlsbeschreibung), mit dem der in der Registerspeicherzelle $0xA5 (Tabelle GL-5.3, „Wert nach Befehlsausführung“) gegebene Wert erreicht werden kann.

	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	…
	POP	RESUME		SYNC	SWYM	

Tabelle GL-5.1: MMIX-Code-Tabelle

Maschinensprache	Assemblersprache	Befehlsbeschreibung
z. B. 0x8D 9B A1 24	*LDO $0x9B, $0xA1, $0x24*	*M[$0xA1 + $0x24] = $0x9B oder: „Lädt den Wert der Datenspeicherzelle an der Adresse A1 plus Offset 24 als Octa in die Registerzelle 9B.“*

Tabelle GL-5.2: Lösungsbeispiel

Nachname, Vorname _____ Matrikelnummer _____

Registerspeicher		
Adresse	Wert vor Befehlsausführung	Wert nach Befehlsausführung
…	…	…
$0xA2	0x00 00 00 00 01 20 00 A9	0x
$0xA3	0x01 02 03 04 05 06 07 08	0x
$0xA4	0x00 00 00 00 00 00 6B 02	0x
$0xA5	0x00 00 00 00 00 00 00 03	0x **0 0 0 0 0 A 5 0 0 0 0 0 0 0 0 0**
…	…	…

Tabelle GL-5.3: Registerspeicher

Datenspeicher																		
Adresse	…	**0x0..0 6B 00**	**0x0..0 6B 01**	**0x0..0 6B 02**	**0x0..0 6B 03**	**0x0..0 6B 04**	**0x0..0 6B 05**	**0x0..0 6B 06**	**0x0..0 6B 07**	**0x0..0 6B 08**	**0x0..0 6B 09**	**0x0..0 6B 0A**	**0x0..0 6B 0B**	**0x0..0 6B 0C**	**0x0..0 6B 0D**	**0x0..0 6B 0E**	**0x0..0 6B 0F**	…
Wert vor Befehls-ausführ.	…	**0x00**	**0x00**	**0x00**	**0x01**	**0x23**	**0x45**	**0x67**	**0x89**	**0xAB**	**0xCD**	**0xEF**	**0x00**	**0x00**	**0x00**	**0x00**	**0x00**	…
Wert nach Befehls-ausführ.	…																	…

Tabelle GL-5.4: Datenspeicher

Maschinensprache	Assemblersprache	Befehlsbeschreibung
0x1D A2 A4 02		
		M_4[$0xA4 + 0x09] = $0xA3
0x84 A4 A5 A4		

Tabelle GL-5.5: Maschinensprache – Assemblersprache – Befehlsbeschreibung

Nachname, Vorname _____ Matrikelnummer _____

Aufgabe BS: Betriebssysteme

Aufgabe BS:
48 Punkte

1. Scheduling

Sechs Prozesse (P1 bis P6) sollen mit einem Einkernprozessor abgearbeitet werden. Das Diagramm BS-1.1 zeigt die Zeiten, zu denen die Prozesse am Einkernprozessor eintreffen und die Ausführungszeiten der einzelnen Prozesse. Prozess 1 tritt periodisch dreimal auf. Die Prozesse sollen zur Laufzeit mit unterschiedlichen Schedulingverfahren eingeplant werden. Alle Schedulingverfahren beginnen zum Zeitpunkt t = 0T. Für die Schedulingverfahren, bei denen feste Prioritäten berücksichtigen werden müssen, ist in der Tabelle BS-1.2 die entsprechende Prioritätenverteilung (Prioritäten 1 bis 6) gegeben.

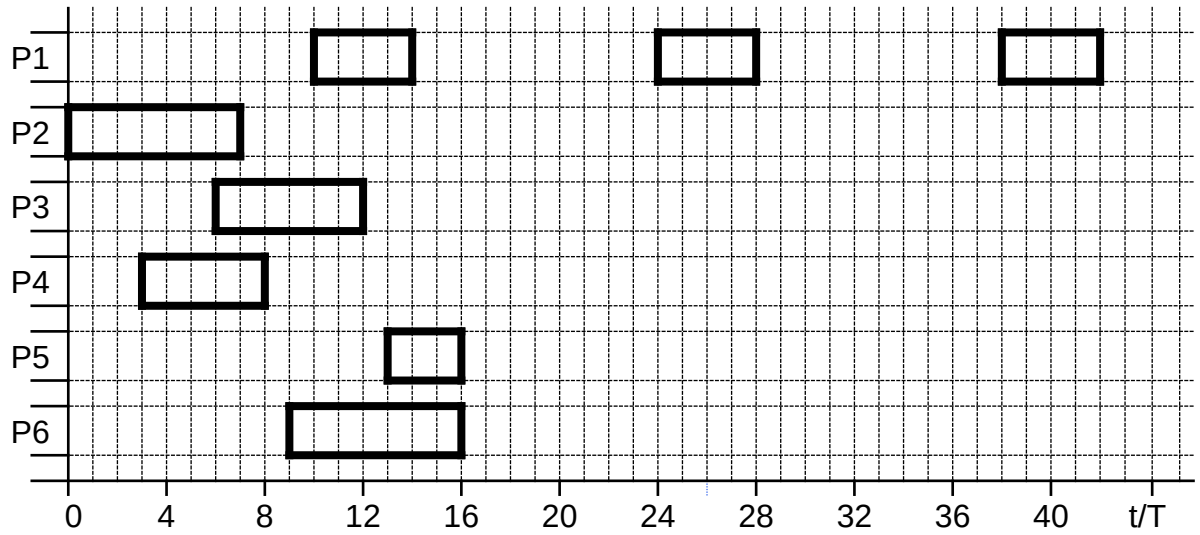


Diagramm BS-1.1: Sollzeitverlauf der Prozesse P1 bis P6

Prozess	P1	P2	P3	P4	P5	P6
Priorität	1 (hoch)	2	3	4	5	6 (niedrig)

Tabelle BS-1.2: Prioritätenverteilung

Hinweis: Bitte füllen Sie die nachfolgenden Aufgaben nach dem folgenden Schema aus:

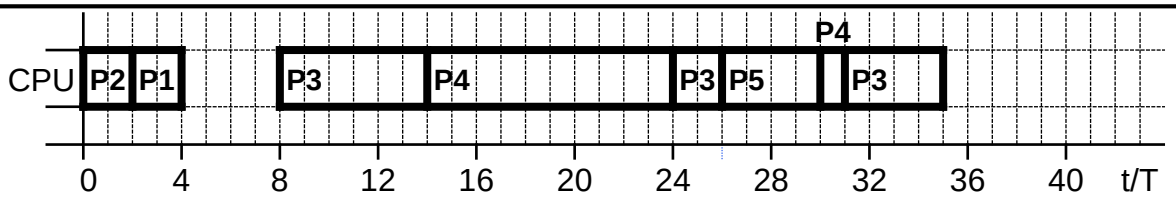


Diagramm BS-1.3: Lösungsbeispiel

Nachname, Vorname _____ Matrikelnummer _____

a) In der ersten Teilaufgabe sollen die Prozesse nicht-präemptiv nach ihren Prioritäten eingeplant werden.

Tragen Sie den Verlauf der Prozessabarbeitung in das Diagramm BS-1.a ein.

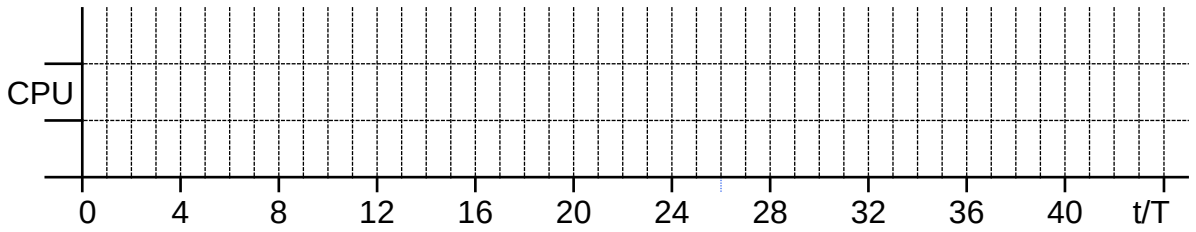


Diagramm BS-1.a: Prioritäten-Verfahren (Prio)

b) In der zweiten Teilaufgabe soll die Einplanung der Prozesse mit dem Round-Robin-Verfahren erfolgen. Neue Prozesse werden nach ihrer Priorität (RR mit Prio) auf die Zeitscheibe gelegt. Für die Zeitscheiben soll angenommen werden, dass diese unendlich viele Schlitze besitzen und so zu jedem Zeitpunkt ausreichend freie Schlitze vorhanden sind. Die Zeitschlitze besitzen eine Länge von 4T. Restzeiten können nicht übersprungen werden.

Tragen Sie den Verlauf der Prozessabarbeitung in das Diagramm BS-1.b ein.

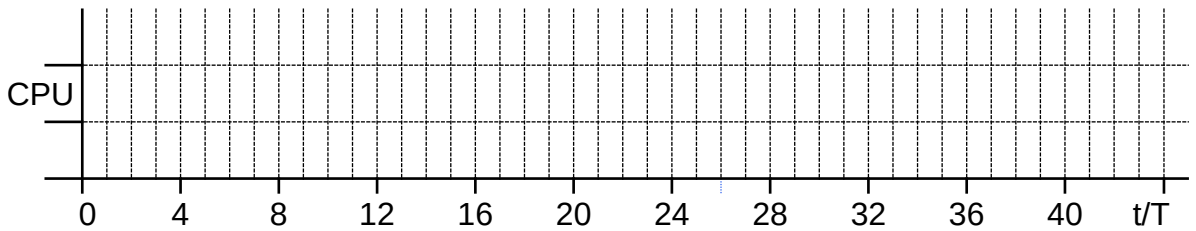


Diagramm BS-1.b: Round-Robin-Verfahren nach Prioritäten (RR mit Prio)

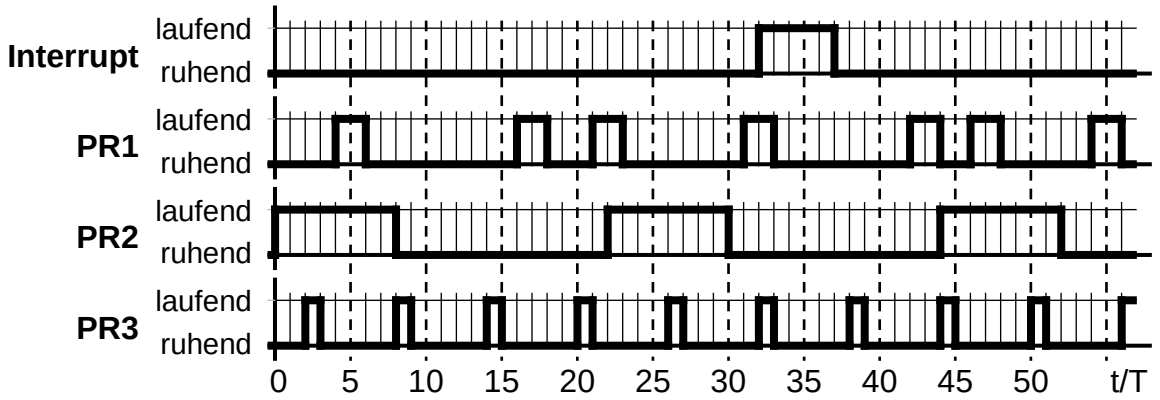
Nachname, Vorname _____ Matrikelnummer _____

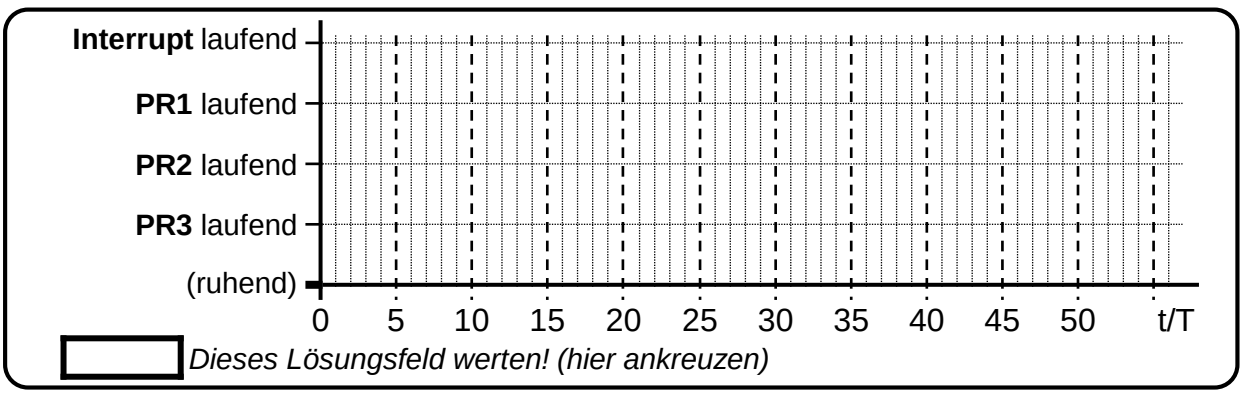
2. Asynchrone Programmierung

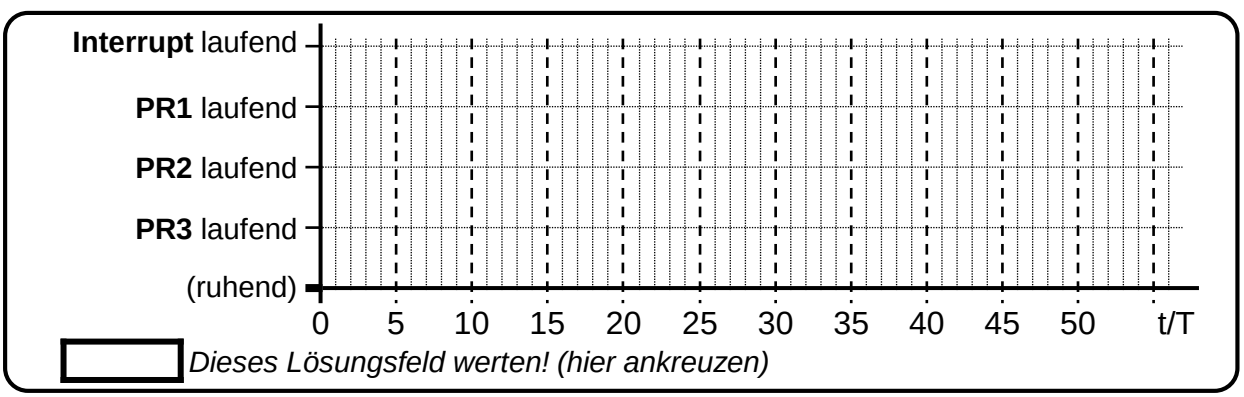
Die zwei periodischen Prozesse PR2 und PR3 sowie der asynchrone Prozess PR1 sollen mit dem Verfahren der asynchronen Programmierung präemptiv auf einem Einkernprozessor eingeplant werden. Der Prozess PR1 besitzt die höchste, der Prozess PR3 die niedrigste Priorität. Die Ausführung wird durch einen Interrupt unterbrochen.

Tragen Sie in das unten angegebene leere Diagramm den Verlauf der Abarbeitung von Programmen und Interrupts ein.

Hinweis: Bei größeren Korrekturen verwenden Sie bitte das Ersatzfeld und markieren das zu wertende Lösungsfeld. Die Bewertung des Verlaufs erfolgt zeilenweise für jeden Prozess/Interrupt und unabhängig voneinander für die erste und zweite Hälfte des Diagramms.



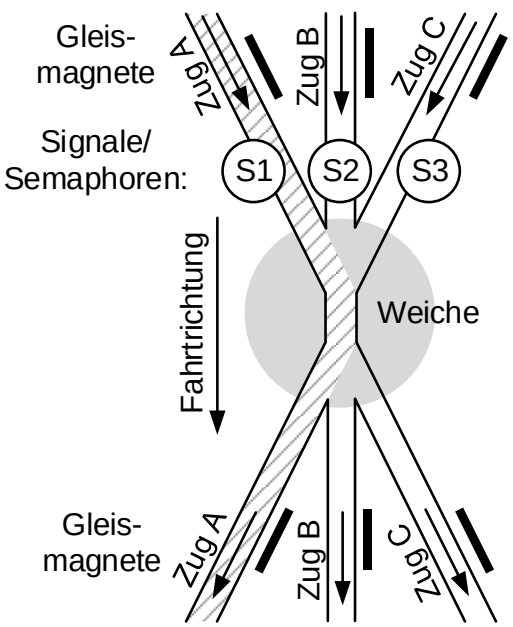




Nachname, Vorname _____ Matrikelnummer _____

3. Semaphoren

Eine Weiche soll die Durchfahrt von drei Zugklassen (Zug A, Zug B, Zug C) durch eine Engstelle regeln. Da die Zugklasse A (Zug A) doppelt so häufig verkehrt wie die Zugklasse C (Zug C) und die Zugklasse B (Zug B) nur einmal verkehrt, soll die Weiche die Züge in folgender Reihenfolge passieren lassen: $\overline{ACCBCC}$.



Entwickeln Sie mit Hilfe der drei Semaphoren S1, S2 und S3 für jede der drei Tasks (Züge) eine Anordnung von Semaphoroperationen. Geben Sie auch Initialwerte für die drei Semaphoren an.

Hinweis: Die Taskreihenfolge muss durch die Semaphoroperationen eindeutig festgelegt sein. Die für die Ausführung eines Tasks notwendigen Semaphoren sollen nur im Block verwendet werden. Beispielsweise würde ein Task X mit folgenden Semaphoroperationen

Task	X
Semaphoren-operationen ↓	P(S7)
	P(S7)
	P(S8)
	…
	V(S9)

nur starten, wenn alle drei benötigten Semaphoren (S7, S7, S8) gleichzeitig frei sind.

Semaphore:	**S1**	**S2**	**S3**
Initialwert:			

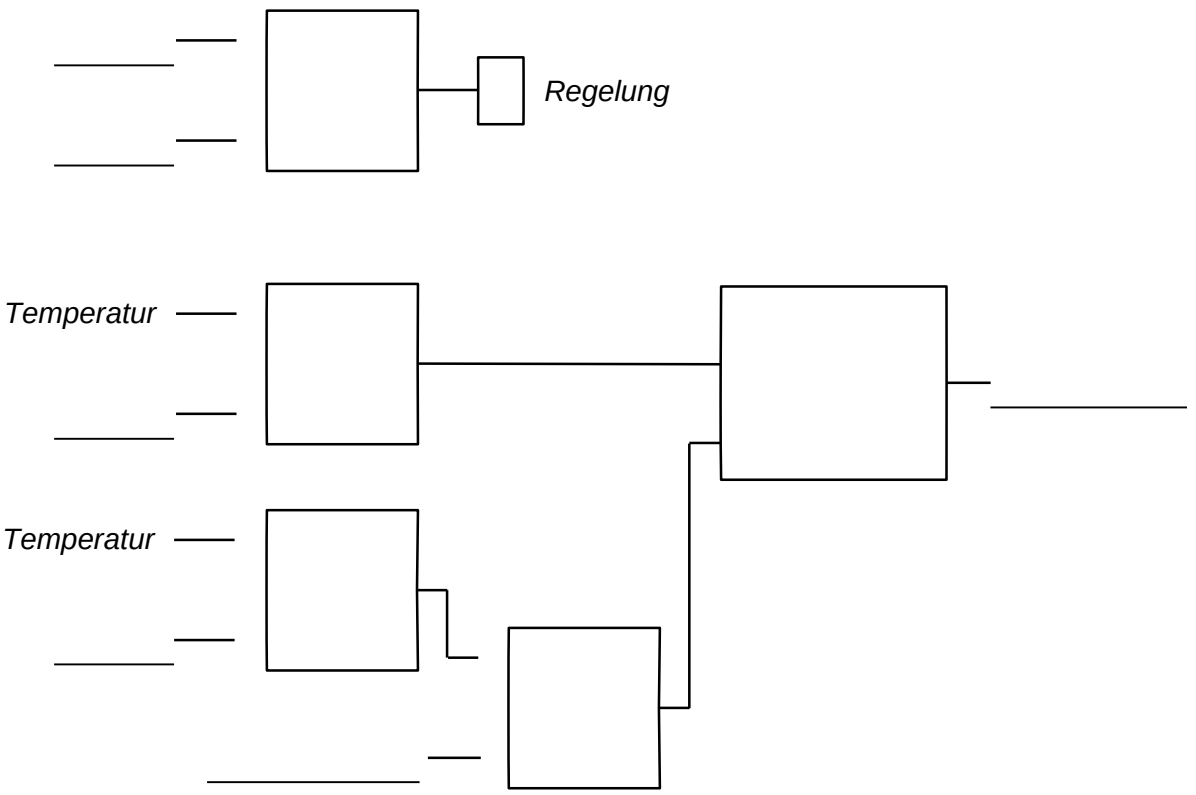
Task:	**A**	**B**	**C**
Semaphor-operationen ↓			

zu definierende Folge der Zugklassen: $\overline{ACCBCC}$

Nachname, Vorname _____ Matrikelnummer _____

4. IEC 61131-3: Funktionsbausteinsprache (FBS)

Eine Trockenanlage besitzt eine *Heizung* und eine *Regelung*, welche die *Heizung* steuert. Zum Aktivieren der *Regelung* müssen zwei Taster (*Taster1*, *Taster2*) gleichzeitig (im gleichen Takt) gedrückt werden. Fällt die *Temperatur* in der Anlage unter *220* Grad, wird die *Heizung* aktiviert; ab *250* Grad und darüber wird die Heizung deaktiviert. Ist die *Regelung* nicht aktiv, muss aus Sicherheitsründen immer auch die *Heizung* abgeschaltet sein.



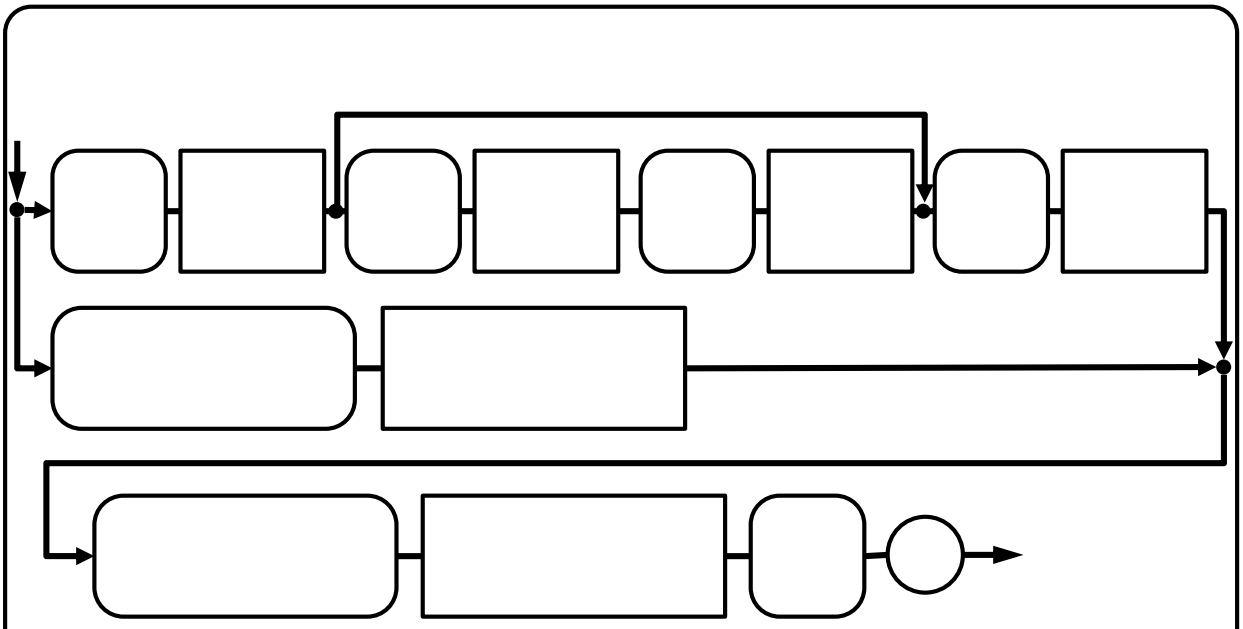
Nachname, Vorname _____ Matrikelnummer _____

5. PEARL

Eine Steuerung führt folgende drei PEARL-Schleifen aus (Variablen i, j und temp seien vorab als Integer mit Wert „1“ initialisiert):

```
FOR i TO 10                          /* Ausleseschleife */
REPEAT
   Auslesen_Sensor(i)
END ;


FOR j FROM 1 BY 2 TO 9               /* Schreibeschleife */
REPEAT
   Schreiben_Sensorpaare(j,j+1)
END ;


WHILE temp<25                        /* Temperaturüberwachung */
REPEAT
   Auslesen_Temperatur(temp)
END ;
```

Leiten Sie aus dem gegebenen Programmausschnitt die Schlüsselwörter, Bezeichner und Syntaxelemente des Syntaxgraphen für Schleifen in der Programmiersprache PEARL ab und tragen Sie diese in die untenstehenden Platzhalter ein.



Legende

SCHLÜSSELWORT	Verzweigung/Zusammenführung
Bezeichner/Ausdruck	Reihenfolge
	elementares Syntaxelement, z. B. Sonderzeichen

Nachname, Vorname _____ Matrikelnummer _____

Aufgabe MSE Teil 1: Automaten und Zustandsdiagramm

Aufgabe MSE1:
20 Punkte

1. Automaten

Gegeben sei folgender unvollständiger Automat (Abbildung 1).

a) Bestimmen Sie ob es sich um einen Moore oder einen Mealy-Automat handelt.

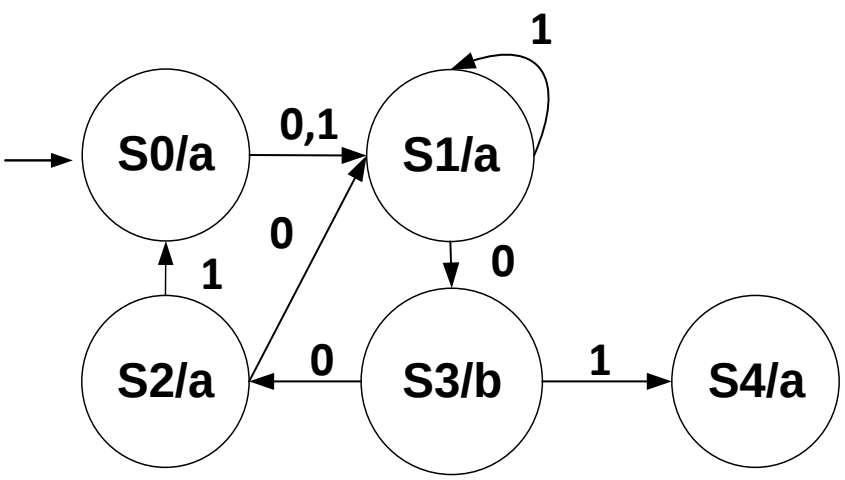
Art des Automaten: _____

b) Welche Ausgabe liefert der Automat wenn die Eingabe **1 1 0 0** erfolgt? In welchem Zustand befindet sich der Automat nach der Eingabe?

Eingabe	Ausgabe	Zustand nach Eingabe
1 1 0 0		

c) Erweitern Sie den Automaten durch zusätzliche Transitionen und ggf. Zustände (benennen Sie diese ggf.) so, dass er sich (ausgehend vom Eingang) nach Eingabe von *0 0 1 1 0* in *Zustand S0* befindet bzw. nach Eingabe von *1 0 1 0 1* in *Zustand S1*. In beiden Eingabefällen sollte die Ausgabesequenz *a a b a b a* lauten.

Abbildung 1

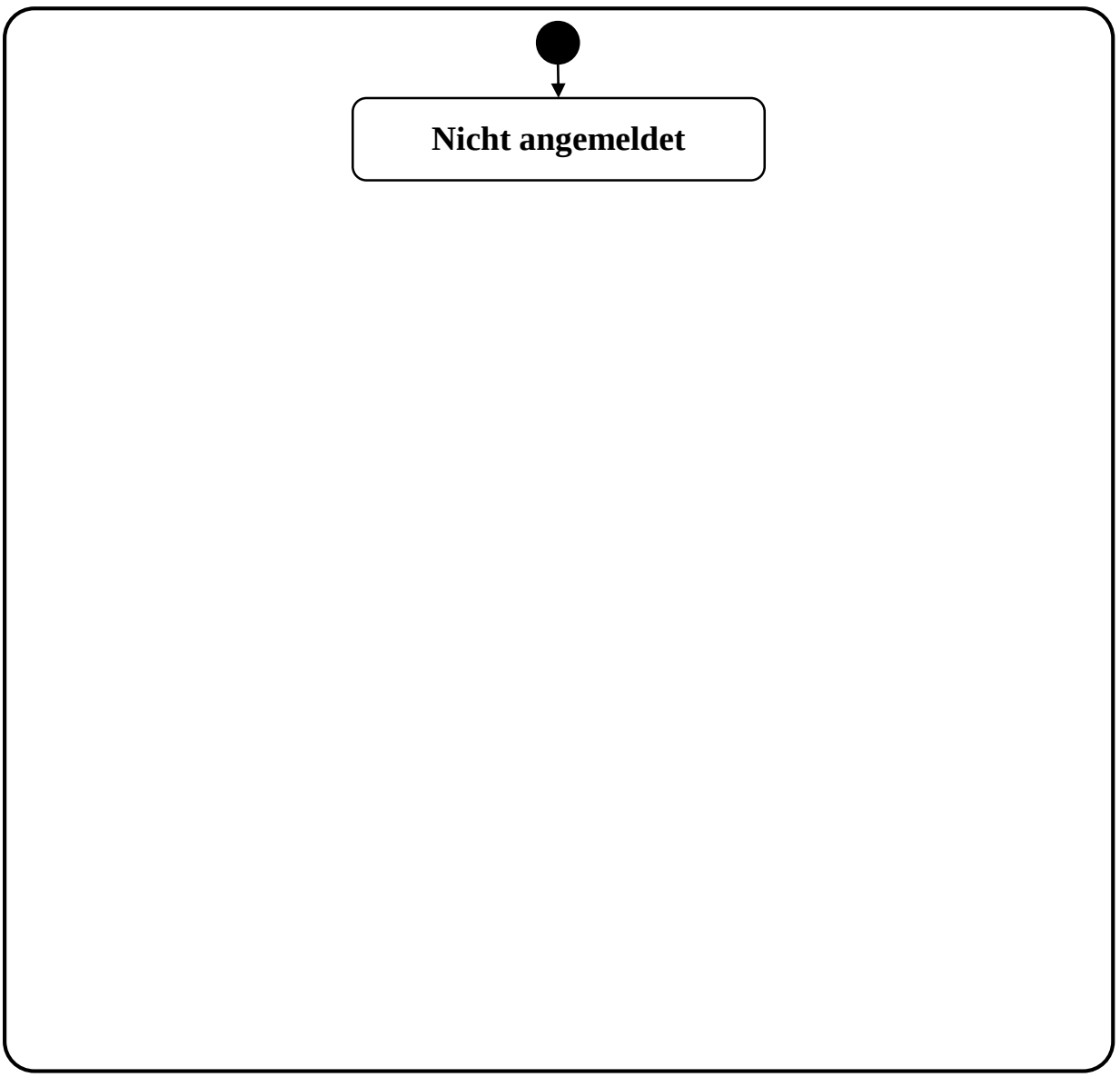


Nachname, Vorname

Matrikelnummer

2. Zustandsdiagramm

Der (vereinfachte) Funktionsablauf eines Geldautomaten soll als Zustandsdiagramm modelliert werden. Nach dem Start befindet sich der Geldautomat im Zustand *Nicht angemeldet*. Durch den Trigger *EC-Karte* geht er in den Zustand *PIN-Abfrage* über und führt beim Betreten des Zustands die Aktion *ErfragePIN* durch. Durch Eingabe der PIN (Trigger *PIN-Eingabe*) wechselt der Automat bei richtiger PIN (Variable *PIN* ist *TRUE*) in den Zustand *Geld-Abfrage*. Bei falscher PIN (Variable *PIN* ist *FALSE*) führt der Automat die Aktion *KarteAusgeben* durch und geht wieder in den Zustand *Nicht angemeldet* über.
Beim Betreten des Zustandes *Geld-Abfrage* führt der Geldautomat die Aktion *ErfrageGeld* aus. Durch den Trigger *Geldsumme* wechselt er dann in den Zustand *Geld-Ausgabe*. Während er sich in diesem Zustand befindet führt er die Aktion *ScheineAusgeben* durch. Wenn die Ausgabesumme korrekt ist (Variable *Ausgabesumme* gleich *TRUE*), wird ebenfalls die Karte ausgegeben und in den Zustand *Nicht angemeldet* gewechselt. Vervollständigen Sie das Zustandsdiagramm entsprechend den Angaben.



Nachname, Vorname

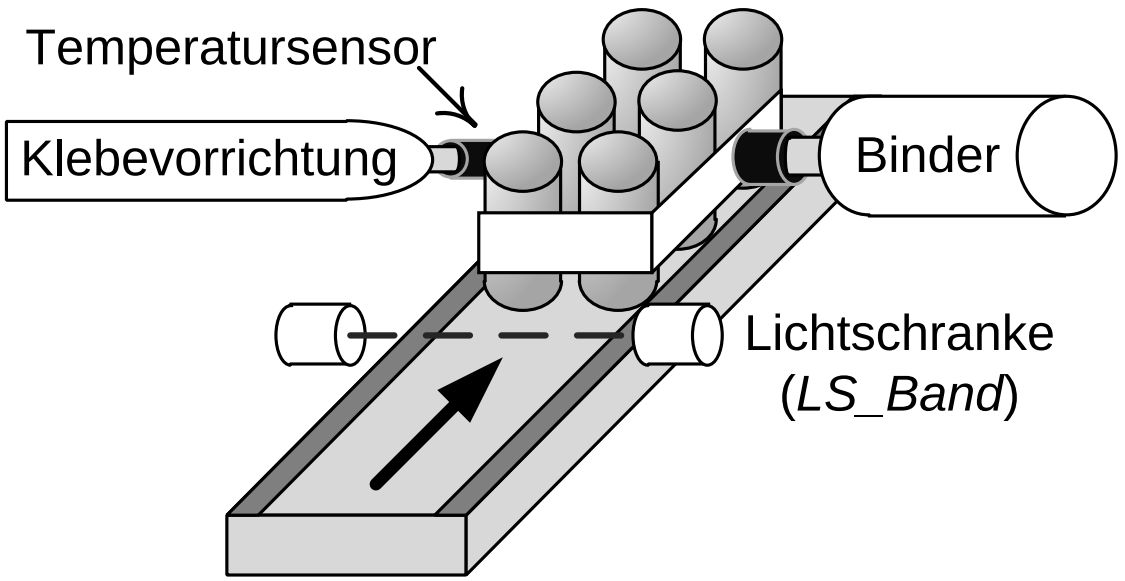
Matrikelnummer

Aufgabe MSE Teil 2: Strukturierte Analyse/ Real-Time (SA/RT)

Aufgabe MSE2:
28 Punkte

Gegeben ist ein Verpackungssystem, in welchem jeweils 6 Dosen zu einem 6-Pack verpackt werden. Die auf dem Transportband ankommenden Dosen werden von einer Lichtschranke (*LS_Band*) erkannt. Wenn 6 Dosen vorhanden sind, werden sie durch den Binder zusammengebunden. Anschließend verklebt eine Klebevorrichtung, an welcher ein Temperatursensor die Klebertemperatur misst, den fertigen 6-Pack.

Das Verpackungssystem soll in den folgenden Aufgaben mittels Strukturierter Analyse/ Real-Time (SA/RT) modelliert werden.



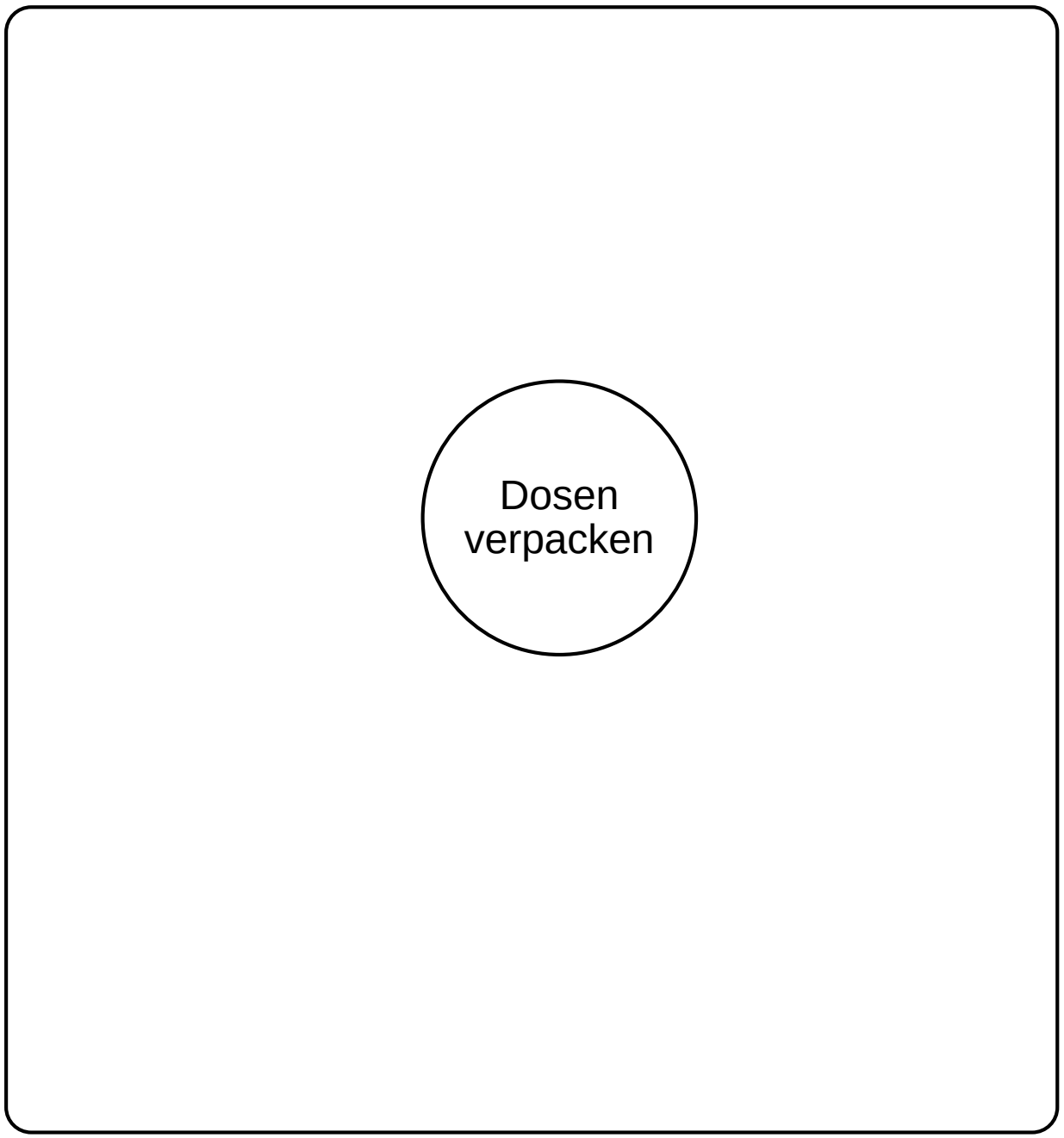
Nachname, Vorname

Matrikelnummer

1. Kontextdiagramm

Es soll das Kontextdiagramm (Datenkontext- und Steuerkontextdiagramm in einem Diagramm) des Prozesses *Dosen verpacken* modelliert werden. Der *Anlagenbediener* erteilt dabei *Kommandos* zur Prozesssteuerung. Vom *Abfüller*, ein dem Prozess vorgelagertes System, kommen die *Dosen*. Die *SPS* (Speicherprogrammierbare Steuerung) liefert dem Prozess die zur Berechnung notwendigen *Sensorwerte* und empfängt die berechneten *Aktorwerte*. Ein dem Verpackungssystem nachgelagerter *Palettierer* empfängt den fertig verpackten *6-Pack* Dosen vom Prozess.

Vervollständigen Sie das unten angegebene Kontextdiagramm gemäß den Vorgaben.



Nachname, Vorname

Matrikelnummer

2. Datenflussdiagramm

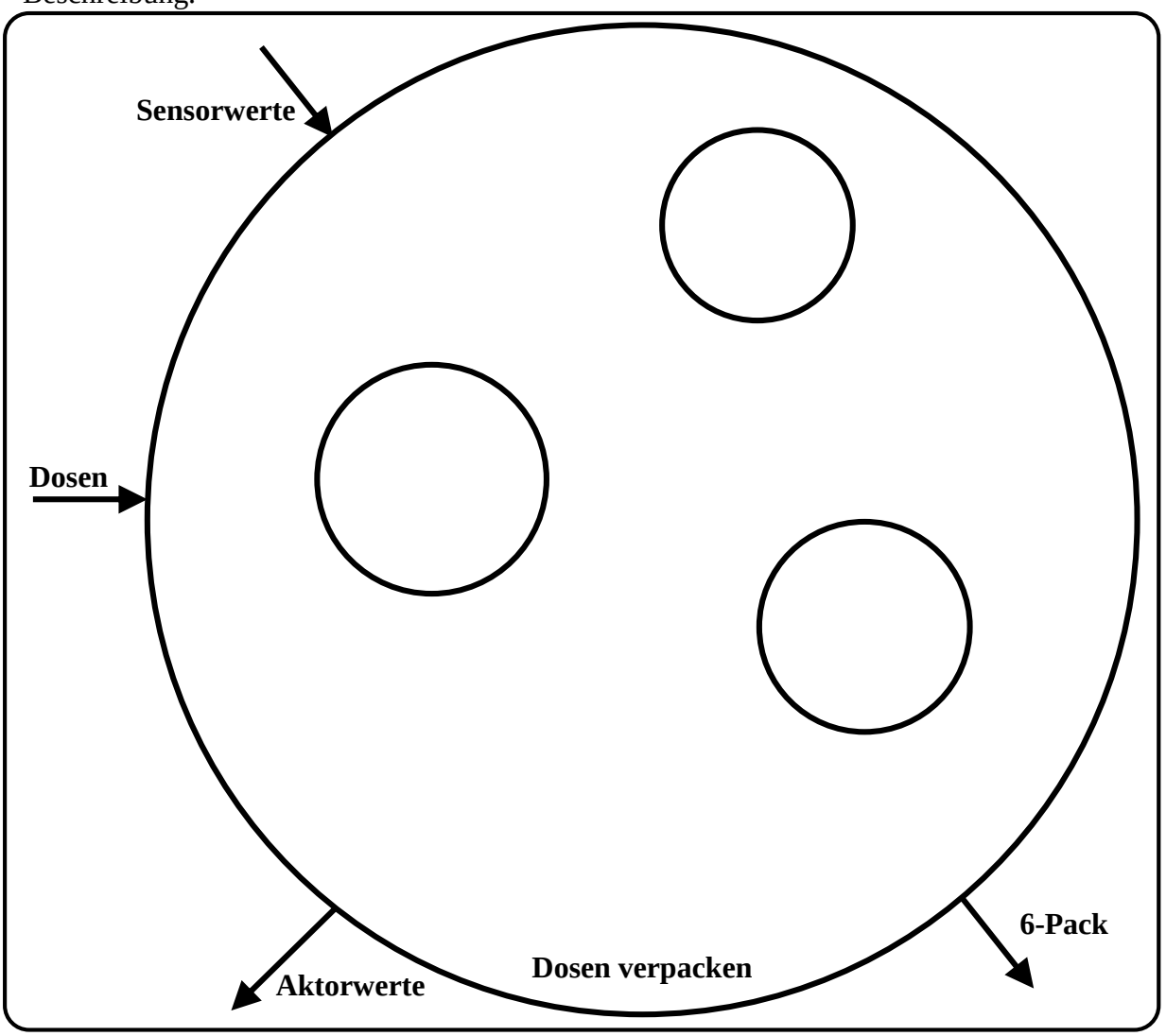
Es soll nun das Datenflussdiagramm des Prozesses *Dosen verpacken* modelliert werden. Der Prozess besteht aus den Subprozessen *Dosen sammeln, Dosen binden, 6-Pack verkleben*. Folgende Flüsse sollen dabei betrachtet werden:

Materialfluss: Am Anfang des Prozesses befinden sich die Dosen auf dem Band (*Dosen auf Band*). Die *gesammelten Dosen* werden dann zusammengebunden. Die *gebundenen Dosen* werden daraufhin zu einem 6-Pack verklebt. Damit ist der 6-Pack zum anschließenden palettieren positioniert (*6-Pack positioniert*).

Sensordaten: Von dem übergeordneten Prozess fließen die *Sensordaten der Lichtschranke* (notwendig zum Sammeln der Dosen) und des *Temperatursensors* (notwendig zum Verkleben) zu den entsprechenden Subprozessen.

Aktordaten: Die Aktordaten setzen sich aus den *Aktordaten des Binders* (zum Binden der Dosen) und den *Aktordaten der Klebevorrichtung* (zum Verkleben des 6-Packs) zusammen.

Vervollständigen Sie das untenstehende Datenflussdiagramm entsprechend der Beschreibung.



Nachname, Vorname _____ Matrikelnummer _____

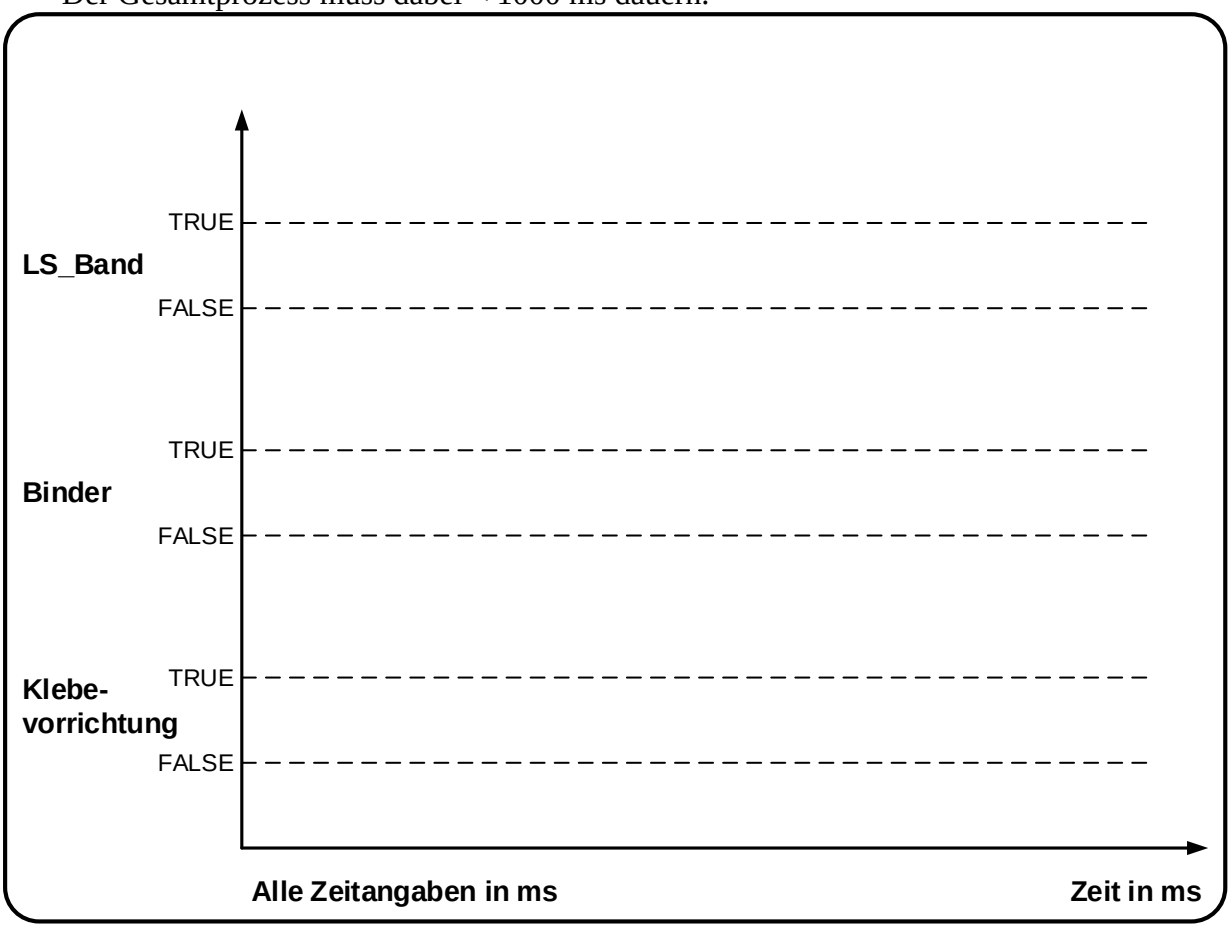
3. Antwortzeitspezifikation: Timing-Diagramm

Zur Verfeinerung des Prozesses *Dosen verpacken* soll nun eine Antwortzeitspezifikation in Form eines Timing-Diagramms durchgeführt werden, um sicherzustellen, dass der Verpackungsvorgang korrekt durchgeführt wird.

Die Werte der Sensoren bzw. Aktoren können jeweils *TRUE* (z.B. Dose erkannt) oder *FALSE* (z.B. keine Dose erkannt) sein.

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

- Sobald die Lichtschranke *LS_Band* die sechste Dose erkannt hat (Dosen 1 – 5 werden nicht betrachtet) muss innerhalb von 300 +/− 10 ms der Binder anfangen, die Dosen zu binden.
- Aufgrund der Breite der Dose liefert die Lichtschranke für 150 +/− 5 ms den Wert TRUE.
- Der Binder braucht für seinen Arbeitsschritt 350 +/− 20 ms und wird anschließend wieder abgeschaltet.
- 100 +/− 20 ms nach Beendigung des Bindevorgangs startet die Klebevorrichtung mit dem Verkleben.
- Der Gesamtprozess muss dabei < 1000 ms dauern.



Nachname, Vorname _____ Matrikelnummer _____

Aufgabe C: Programmieren in C

Aufgabe C:
96 Punkte

1. Datentypen, Ein-/Ausgabe und boolesche Algebra

a) Erstellen Sie die Definition der folgenden Variablen mit Datentypen, so dass durch das Ablegen **so wenig** Speicher wie möglich genutzt wird. Wählen Sie aussagekräftige Variablenbezeichnungen, um Daten über ein Werkstück speichern zu können.

- RFID-Werkstücknummer (10-stellige positive Zahl, max. jedoch 4.000.000.000)
- Werkstücktyp (100 verschiedene Typen im Sortiment)
- Schichtdicke der aufgetragenen Lackierung auf zwei Nachkommastellen genau

Geben Sie zudem die Variable der Schichtdicke mittels formatierter Ausgabe ohne ihre Nachkommastelle und ohne zu Runden aus (Nachkommastelle wird abgeschnitten).

```
_____________________________________________

_____________________________________________

_____________________________________________

// Formatierte Ausgabe der ermittelten Schichtdicke

_________(__Schichtdicke: _____________________)__
```

b) Konstruieren Sie einen booleschen Ausdruck (kein ablauffähiger Code benötigt) welcher die Variable *iJahr*, die eine Jahreszahl speichert, darauf bewertet ob das Jahr ein Schaltjahr ist oder nicht. Der Ausdruck soll eine logische „1“ (TRUE) zurückgeben, falls das Jahr ein Schaltjahr ist und eine „0“ (FALSE) falls nicht. Für Schaltjahre gilt:

- Alle 4 Jahre findet ein Schaltjahr statt (d.h. die Jahreszahl ist durch 4 teilbar)
- Alle 100 Jahre wird das Schaltjahr ausgesetzt.
- Alle 400 Jahre findet trotzdem ein Schaltjahr statt.

Nachname, Vorname _____ Matrikelnummer _____

2. Kontrollstrukturen

Ein Programm zur automatischen Qualitätssicherung soll die Schichtdicken von zwei produzierten Werkstücktypen A und B prüfen. Dazu werden zum einen die Werkstücktypen im char-Array *cTyp* und an derselben Stelle im float-Array *fMesswert* die Schichtdicken abgelegt. Beide Arrays besitzen die Größe MAX, welche mittels eines Präprozessor-Befehls gesetzt ist. Schreiben Sie nun ein C-Programm welches mithilfe der sog. *fabs()*-Funktion zur Berechnung des Absolutwertes (Bsp.: fabs(-2.4) gibt die Zahl +2.4 zurück) die Schichtdicken auf Einhaltung der Toleranz des jeweiligen Typs prüft. Ist die Toleranz nicht eingehalten, soll die Warnmeldung „Werkstück ... ist fehlerhaft!“ausgegeben werden.

- Werkstück Typ A besitzt die Solldicke *fsoll_A* mit einer Toleranz von +/- *ftol_A*
- Werkstück Typ B besitzt die Solldicke *fsoll_B* mit einer Toleranz von +/- *ftol_B*

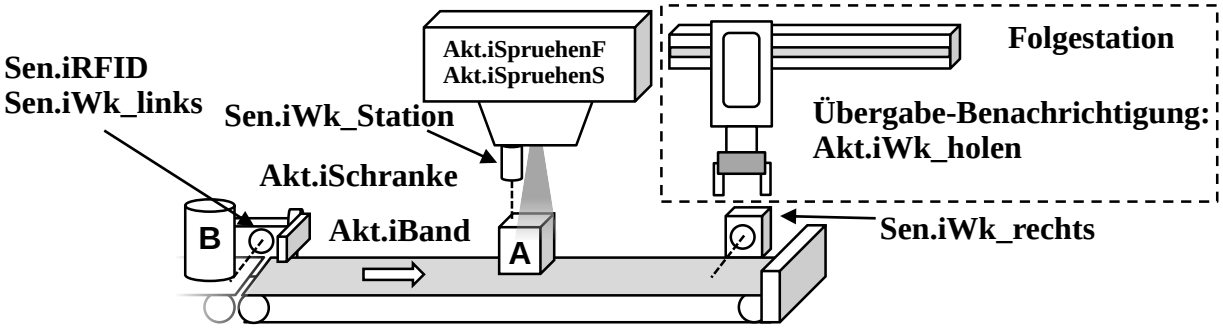
Lassen Sie weiterhin die Anzahl von fehlerhaften Werkstücken in der Variable *iZaehler* zählen und von der Funktion als Rückgabewert zurückgeben.

```
#include <stdio.h>
#include <math.h>
#define MAX 5
int QualitaetsMessung()
{
    float ftol_A=0.09, ftol_B=0.12;
    float fsoll_A=2.5, fsoll_B=3.8;
    float fMesswert[MAX]  = {2.6,3.81,3.4,2.58,3.8};
    char cTyp[MAX]={'A','B','B','A','B'};
    int   i=0, iZaehler=0;
```

```
}
```

Nachname, Vorname _____ Matrikelnummer _____

3. Zyklische Programmierung einer Lackieranlage



Die Firma ISA GmbH stellt einen vollautomatisierten Lackierautomaten her, welcher zwei Typen von Werkstücken verarbeiten kann. Die Anlage nimmt stets nur ein Werkstück entgegen, liest den Typ über einen RFID-Tag aus und positioniert es dann unter einer Sprühdüse. Gerade IDs werden als Typ A (nur Farbe sprühen) und ungerade IDs als Typ B (Farbe und Schutzlack sprühen) identifiziert. Der Ablauf und die Übergangsbedingungen der zugehörigen Steuerung ist auf der folgenden Seite in einem Zustandsdiagramm beschrieben.

Zustand 0: Um in den Zustand 1 zu wechseln wird ein neues Werkstück erkannt und die RFID in iRFID gespeichert.

Zustand 1: Das Band wird aktiviert und die Schranke für 2 Sekunden geöffnet. Anschließend wird Akt.iSchranke wieder zurückgesetzt. Wird die Ankunft an der Sprühstation erkannt (Dauer deutlich über 2 Sekunden), kann nach Zustand 2 gewechselt werden.

Zustand 2: Das Band wird angehalten und anschließend das Auftragen der Farbe für 4.5 Sekunden aktiviert. Anschließend wird anhand der gespeicherten RFID entschieden ob noch Schutzlack aufgetragen wird (Zustand 3) oder abtransportiert wird (Zustand 4).

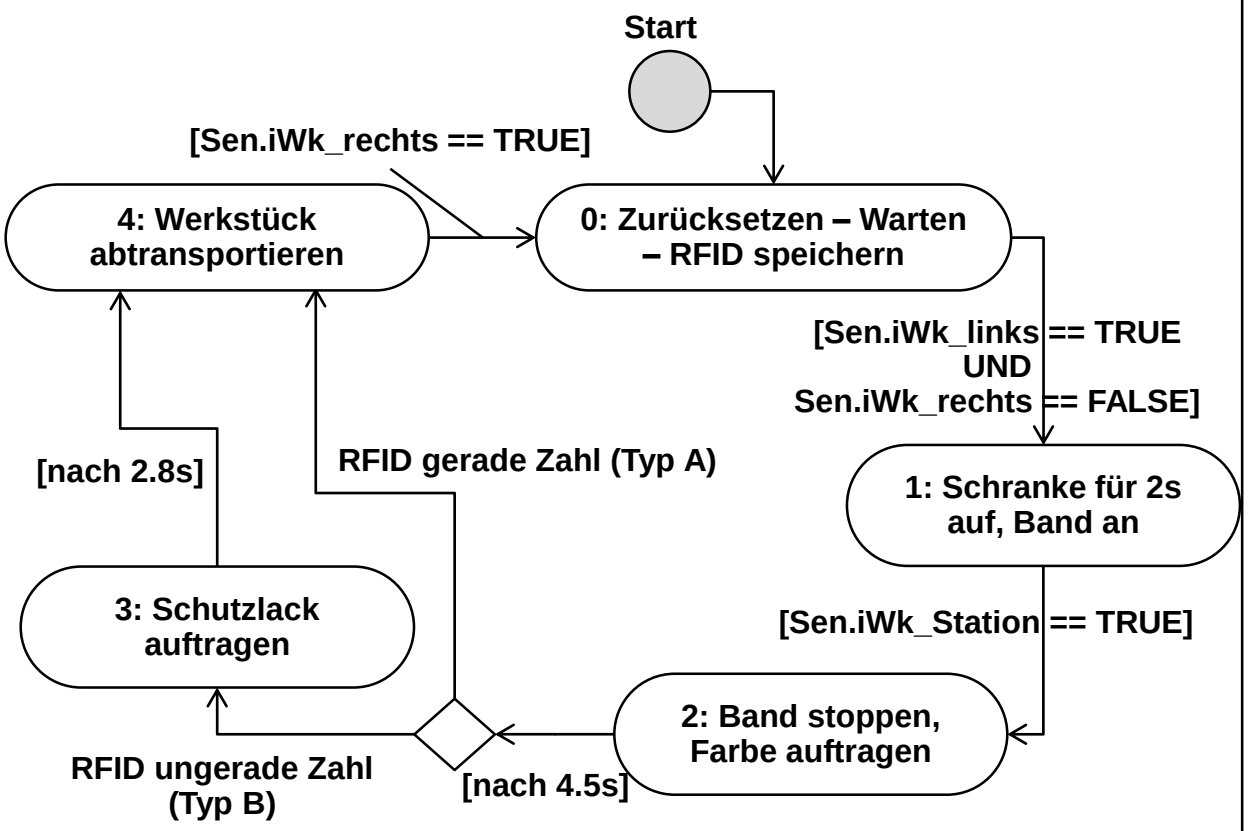
Aktoren:	
Akt.iSchranke	1: Öffnet / 0: Schließt die Schranke zum Einlassen eines neuen Werkstück
Akt.iBand	Startet den Bandmotor
Akt.iSpruehenF	Aktiviert die Sprühdüse mit Farbe
Akt.iSpruehenS	Aktiviert die Sprühdüse mit Schutzlack
Akt.iWk_holen	Meldet der Folgestation, dass ein fertiges Werkstück geholt werden kann
Sensoren/Merkvariablen:	
Sen.iRFID	RFID-Sensor an der Schranke zum Auslesen der Werkstück ID
Sen.iWk_links	Sensor erkennt Vorhandensein eines Werkstücks an der Schranke
Sen.iWk_Station	Sensor erkennt Vorhandensein eines Werkstücks an der Sprühstation
Sen.iWk_rechts	Sensor erkennt Vorhandensein eines Werkstücks am Ende des Bandes
t	Speichert eine Zeit (in ms) für die Verwendung als Timer
iRFID	Zwischenspeicher für den ausgelesenen RFID-Wert
plcZeit	Systemlaufzeit der Steuerung (in ms)
state	Speichert den aktuellen Zustand

Nachname, Vorname | Matrikelnummer

Zustand 3: Der Schutzlack wird nun für 2.8 Sekunden aufgetragen. Anschließend wird die Düse zurückgesetzt und zum Zustand 4 gewechselt um das fertige Werkstück vom Typ B abzutransportieren.

Zustand 4: Zum Abtransport wird wieder das Band gestartet und angehalten sobald das Werkstück auf der rechten Seite angekommen ist. Anschließend wird der Folgestation mittels eines Impulses (Setzen des Aktors für mindestens einen PLC-Zyklus) über den Aktor *Akt.iWk_holen* signalisiert, dass ein Werkstück fertig ist und wieder zum Zustand 0 gewechselt, um auf ein neues Werkstück zu warten.

Vorgehen: Übertragen Sie das Zustandsdiagramm und die oben beschriebene Funktionalität in den C-Code auf den folgenden Seiten. Benutzen Sie die Variablen auf der vorherigen Seite. Beachten Sie, dass der Code zyklisch ausgeführt wird. Der Kopf des Programms mit Deklaration und Initialisierung aller Variablen ist bereits unten gegeben.




```
#include "eavar_lackieren.h"

struct SData Sen;          // Sensorvariablen
struct AData Akt;          // Aktorvariablen
unsigned int state=0;      // Zustandsvariable (Start in 0)
unsigned int plcZeit=0;    // in ms (1000 entspricht 1 Sek.)

int iRFID; // Speicher für RFID-Wert
int t;     // Speicher für Timer-Messung (in ms)
```

Nachname, Vorname _____ Matrikelnummer _____

```
switch(state)
{
  ____________________ //State 0 Variablen setzen

  Akt.iWk_holen=0; //Zurücksetzen

  if(Sen.iWk_links && !Sen.iWk_rechts)//Trans. 0->1
  {
        ____________________________________

        ____________________________________
  }break;

  ____________________//State 1 Schranke auf, Band an

  ____________________________________// Aktoren setzen

  ____________________________________

  ____________________________________________// Timer

  ____________________________________________// Schranke zu

  ____________________________________________

  ____________________________________________// Trans. 1->2

  {     ____________________________________

        ____________________________________  } break;

  _________________//State 2 Band halt, Farbe auftragen

  ____________________________________// Aktoren setzen

  ____________________________________

  ____________________________________// Timer starten

  ____________________________________

  {
     // Fortsetzung folgende Seite
```

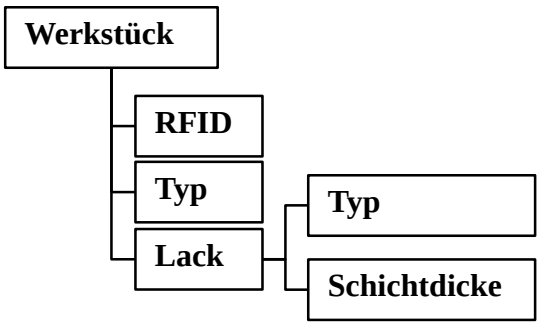
Nachname, Vorname _____ Matrikelnummer _____

```
    _____// Aktor setzen
    _____//Trans. 2->3 o. 2->4
    _____________________________________
    _____________________________________ } break;
_____________//State 3 Schutzschicht auftragen
_____________________________ // Aktoren setzen
_____________________________ // Timer starten
_____________________________ // Transition 3->4
 {
      _____________________________
      _____________________________
      _____________________________
 }break;
_____________//State 4 Abtransport
_____________________________ // Aktoren setzen
_____________________________ // Transition 4->0
 {
     _____________________________ // Aktoren setzen
     _____________________________
     _____________________________
 } break;
```

Nachname, Vorname _____ Matrikelnummer _____

4. Speichern der Werkstücke in eine Datei

Die zuvor mit Hilfe der Lackieranlage verarbeiteten Werkstücke sollen nun an einer weiteren Station geprüft und die Daten in eine Datei zur Qualitätssicherung übernommen werden. Um eine hohe Nachverfolgbarkeit zu gewährleisten, speichert die Steuerung daher alle Daten über die Lackschicht in die Textdatei „messung.txt“.



Um die Daten geordnet in ein C-Programm einzulesen, sollen zunächst zwei Datentypen angelegt werden. Einmal ist dies der Datentyp WERKSTUECK, welcher den Datentyp LACK enthält, um die Daten zur Lackschicht geordnet ablegen zu können. Die Datenstruktur ist oben schematisch abgebildet.

a) Legen Sie einen neuen Datentyp mit dem Namen „LACK“ an und definieren Sie folgende Variablen in der Struktur:

- Lacktyp (2 Zeichen im Format „Zahl Buchstabe“ z.B. 5A)
- Schichtdicke (mind. 3 Nachkommastellen). Verwenden Sie einen möglichst kurzen Variablennamen zur besseren Übersicht in folgenden Teilaufgaben.

```
#include <stdio.h>
#include <string.h>
```

b) Legen Sie einen neuen Datentyp mit dem Namen „WERKSTUECK“ an und definieren Sie folgende Variablen in der Struktur:

- RFID (Zahl ohne Nachkommastellen)
- Werkstücktyp (max. 1 Zeichen)
- Lack vom bereits erstellten Datentyp LACK

Nachname, Vorname _____ Matrikelnummer _____

c) Vervollständigen Sie nun ein Programm zum Abspeichern der Messung. Legen Sie dabei ein Array „wkDB“ vom Typ WERKSTUECK an, welches 20 Einträge speichern kann. Erstellen Sie anschließend die Datei „werkstueck.txt“ mittels eines FILE-Handle mit der Bezeichnung „pSpeichern“ so, dass sie vom Programm neu beschrieben werden kann. Um alle Daten abzuspeichern, ist eine Schleife notwendig, die alle Array-Einträge einzeln übergibt. Verwenden Sie in der Schleife einen geeigneten Befehl, um am Anfang jeder Zeile den String „RFID:“ und dann die aktuelle ID abzuspeichern. Schließen Sie schlussendlich den Dateizugriff nach dem Ende der Schleife wieder.

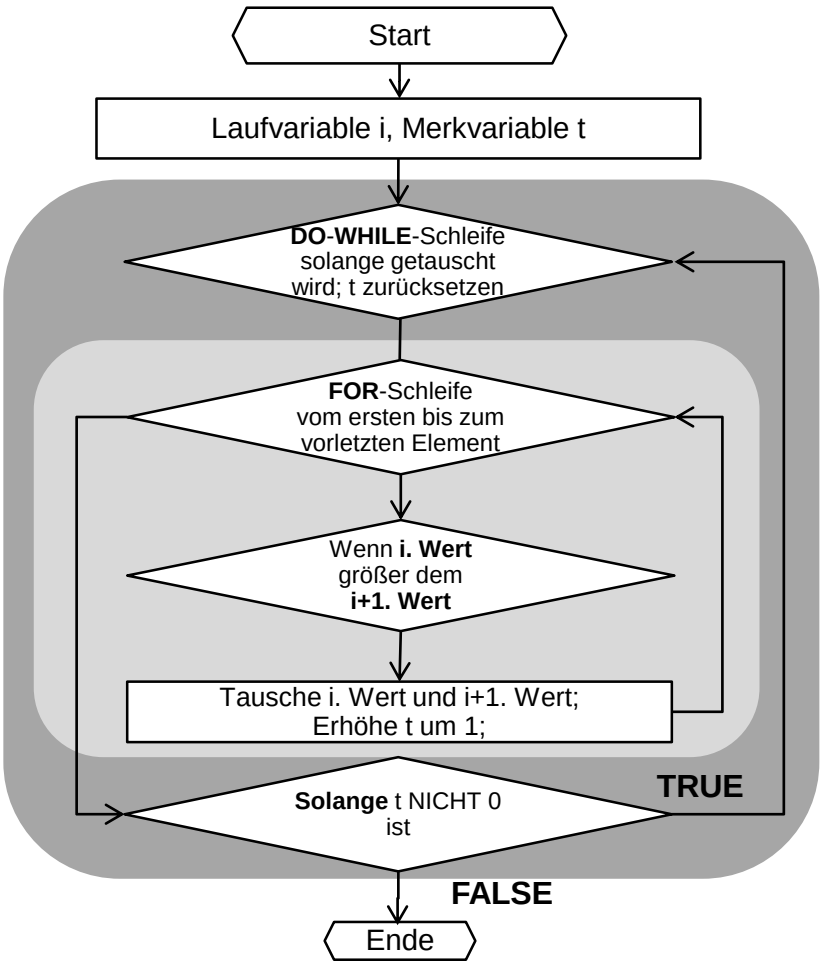
```
 int main()
{   //Variablendeklaration:
    int i=0, j=0;


    ____________________________________________
     wkDB=RFID_readall(); // Einlesen der RFID in die DB
    //Öffnen der Datei werkstueck.txt:


    ____________________________________________________________
    //Schleife vom Anfang bis Ende des Array:


    ____________________________________________________
    {
      ____________________________________________________
    }
    //Schließen der Datei:
    ____________________________________________
    return 0;
}
```

Nachname, Vorname

Matrikelnummer

5. Sortieren mittels Bubble-Sort

Zur Ausgabe der Schichtdicken aller aktuell produzierten Werkstücke soll eine Funktion implementiert werden, welche das Array vom Typ WERKSTUECK mit einer beliebigen Arraygröße „iGroesse“ sortieren kann. Hierbei sollen die Messungen mit der Höhe *aufsteigend* (also dünnste Schicht zuerst) sortiert werden. Hierfür soll der sog. *Bubble-Sort-Algorithmus* implementiert werden, welcher im Folgenden ausführlich in einem Flussdiagram beschrieben ist.



Zum weiteren Verständnis der Wirkungsweise des *Bubble-Sort-Algorithmus* ist im Folgenden ein Beispiel für die Zahlenreihe 1-4-3-2 durchgeführt. Die Plätze im Kästchen werden im aktuellen Schritt verglichen:

[1 4] 3 2	1 größer 4 ? ➔ Nein	→ i++;
1 [4 3] 2	4 größer 3 ? ➔ Ja, Tausch! ➔ 1 3 4 2	→ i++; t++; (t=1)
1 3 [4 2]	4 größer 2 ? ➔ Ja, Tausch! ➔ 1 3 2 4	→ FOR beendet; t++; (t=2) ➔ DO, t=0
[1 3] 2 4	1 größer 3 ? ➔ Nein	→ i++;
1 [3 2] 4	3 größer 2 ? ➔ Ja, Tausch! ➔ 1 2 3 4	→ i++; t++; (t=1)
1 2 [3 4]	3 größer 4 ? ➔ Nein	→ FOR beendet; (t=1) ➔ DO, t=0

➔ Da bereits sortiert, ein FOR-Zyklus ohne Tauschen➔ (t=0) Abbruchkriterium erfüllt

Nachname, Vorname _____ Matrikelnummer _____

Vervollständigen Sie nun unten die Funktion „bubbleSort“ entsprechend der zuvor im Flussdiagramm beschriebenen Funktionsweise. Die Funktion wird mittels *call-by-reference* aufgerufen und bekommt einen Zeiger auf das Array übergeben. Sie sortiert also direkt im Array vom Typ WERKSTUECK und gibt keinen Rückgabewert zurück.

Die verwendete Schleife benötigt die Größe des Arrays als weiteren Parameter, welche durch die int-Variable *iGroesse* an die Funktion per Kopie (*call-by-value*) übergeben wird.

Die Arrayelemente sollen nach der Struct-Variable für die Schichtdicke *aufsteigend* sortiert werden. Diese ist im Datentyp LACK enthalten (z.B. *fDicke*), welcher wiederum Teil des Datentyp WERKSTUECK ist (z.B. *lack*).

Es sei des Weiteren bereits eine Funktion „*tausche*“ implementiert, die nicht dargestellt ist. Die Funktion kann zwei Arrayelemente vom Typ WERKSTUECK vertauschen. Verwenden Sie diese durch einen korrekten Funktionsaufruf mittels *call-by-reference*, indem Sie ihr die beiden Tauschpartner (also die entsprechenden Arraystellen) als Adresse übergeben. **Hinweis:** Beachten Sie die zugehörigen Kommentare im Code.

```
_____________________________________________________
{
  ___________________  // Variablen deklarieren

  ___________________ // Aeussere Schleife
  {

    _____________________________

    _____________________________ //Innere Schleife

    {
      ___________________________________________________

      ___________________________________________________

      _________________________
    }

  } _________________

}
```

Nachname, Vorname _____ Matrikelnummer _____

6. Verkettete Liste

Gegeben ist folgender Ausschnitt eines C-Programms mit einer verketteten Liste, welche wiederum Werkstücke speichert. Ein Werkstück habe hierbei zur Vereinfachung nur seine RFID als Inhalt. Der Code zum Einfügen neuer Elemente ist nicht dargestellt. Vervollständigen Sie die Funktion „Ausgabe“, welche es ermöglicht, ein einzelnes Element der verketteten Listen aufzurufen und zurückzugeben. Der zugehörige Aufruf zum Anzeigen der RFID des 5. Elements wurde in der „main“-Funktion implementiert.

```
(...)
typedef struct LISTE_s{
  int RFID;
  struct LISTE_s *pNext;
}WERKSTUECK;

typedef struct{
  int iAnzahl;
  WERKSTUECK *pFirst;
}KOPF;

_________________ Ausgabe(_________ Liste,_______ iStelle)
{
  int ia=0;
  WERKSTUECK* pTemp=NULL;
// Zeiger umbiegen bis die gesuchte Stelle erreicht ist.

          ____________________________________

          ____________________________________

          {

          ____________________________________

          }
return pTemp;
}
(...)

int main()
{
 //Variablendeklaration
  int i=0; KOPF Liste={0,NULL}; WERKSTUECK *pAusgabe=NULL;

//Ausgabe des 5. Element
    pAusgabe=Ausgabe(&Liste, 5);
    printf("5.Werkstueck RFID: %i\n",pAusgabe->RFID);
(...)
}
```