Lehrstuhl für Informationstechnik im Maschinenwesen
Prof. Dr.-Ing. B. Vogel-Heuser
Prof. Dr.-Ing. F. Schiller

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Sommersemester 2010

27. August 2010

Bitte legen Sie Ihren Lichtbildausweis bereit.

Sie haben für die Bearbeitung der Klausur 120 Minuten Zeit.

Diese Prüfung enthält **25** 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	ZS	BS	DB	MO	RK	∑	**Note:**
erreichte Punkte							
erzielbare Punkte	**54**	**64**	**30**	**50**	**46**	**244**	

Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe ZS: Zahlensysteme und logische Schaltungen

Aufgabe ZS: 54 Punkte

a) Überführen Sie die unten gegebenen Zahlen in die jeweils anderen Zahlensysteme. *Wichtig:* Achten Sie genau auf die angegebenen *Basen*!

(101101)$_2$ (**55**)$_8$ (**45**)$_{10}$ (**2D**)$_{16}$

(**31**)$_4$ (**16**)$_7$ (13)$_{10}$ (**D**)$_{16}$

b) Stellen Sie mit Hilfe der Max- und Minterme die Disjunktive- und Konjunktive Normalform aus der unten angegebenen Wahrheitstabelle auf. A, B und C sind die Eingänge der Funktion, Y ist der Ausgang.

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

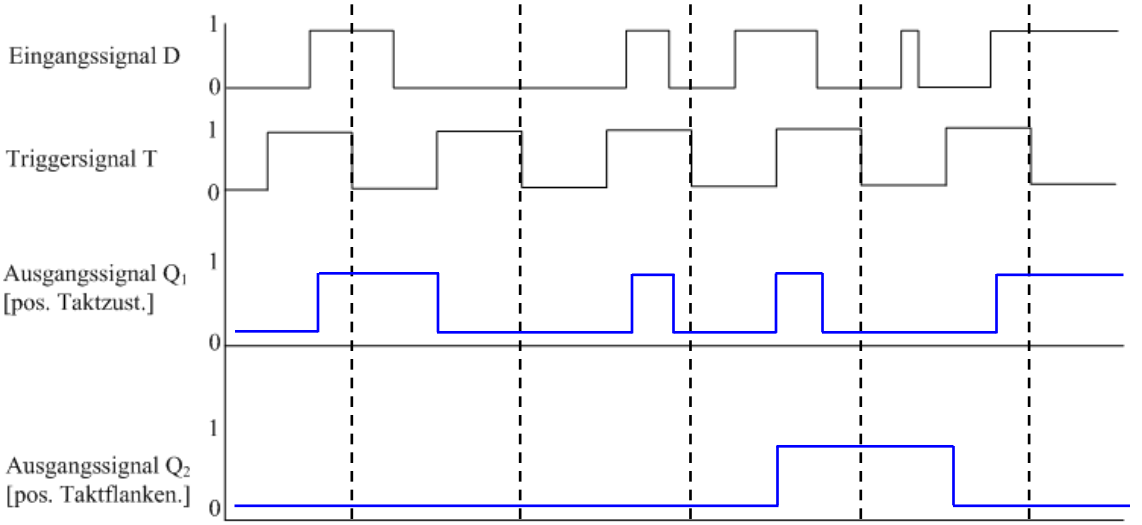
$$Y_{DNF} = (\overline{A} \wedge \overline{B} \wedge \overline{C}) \vee (\overline{A} \wedge \overline{B} \wedge C) \vee (\overline{A} \wedge B \wedge C) \vee (A \wedge B \wedge \overline{C})$$

$$Y_{KNF} = (A \vee \overline{B} \vee C) \wedge (\overline{A} \vee B \vee C) \wedge (\overline{A} \vee B \vee \overline{C}) \wedge (\overline{A} \vee \overline{B} \vee \overline{C})$$

Vorname, Name _____ Matrikelnummer _____

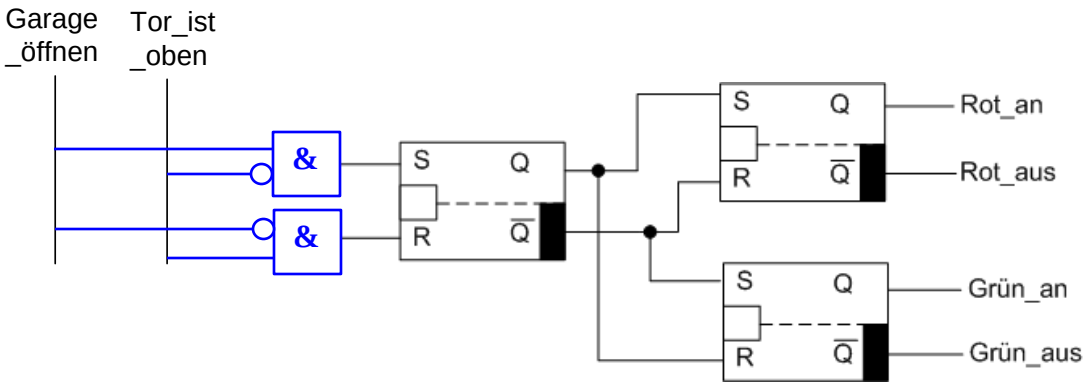
Punkte

c) Gegeben ist das *Eingangssignal D* und das *Triggersignal T* (Clock).
Zeichnen Sie das *Ausgangssignal* für ein *positiv zustandsgesteuertes D-FlipFlop* in Q_1 und für ein *positiv taktflankengesteuertes D-FlipFlop* in Q_2 ein.



d) Die unten angegebene FlipFlop-Schaltung ist für das Öffnen eines Garagentors zuständig (schließen wird nicht betrachtet). Drückt der Fahrer den Knopf „*Garage_öffnen*“ (1 = gedrückt, sonst 0), schaltet die Ampel „*Rot_an*“ und „*Grün_aus*“. Sobald das Garagentor vollständig geöffnet ist, wird automatisch ein Schalter „*Tor_ist_oben*“ aktiviert (1=aktiviert, sonst 0). Dann schaltet „*Rot_aus*“ und „*Grün_an*“.
Beschreiben Sie mit Stichworten die Situation, die ungültige Zustände bei dem RS-FlipFlop auslöst falls S mit Garage_öffnen und R mit Tor_ist_oben verbunden werden. Ergänzen Sie die Schaltung um zwei UND-Gatter, damit diese ungültigen Zustände verhindert werden.

Antwort: Ist „Tor_ist_oben“ UND drückt der Fahrer den Knopf „Garage_öffnen“ liegt 1,1 am FlipFlop Eingang (ungültig).



Vorname, Name _____ Matrikelnummer _____

Punkte

e) Das folgende Programm implementiert die Garagentorsteuerung aus Aufgabenteil d). Auch hier wird nur die Schaltung der Ampel beim Öffnen betrachtet.
Die Sensorwerte liegen in den Variablen `cGarageOeffnen` und `cTorIstOben`, die Ausgänge (0=Aus, 1=An) werden über die Variablen `cRotAn` und `cGruenAn` gesetzt.
Gestalten Sie dass Programm so, dass ein gleichzeitiges Drücken des „Garage öffnen“ Schalters bei vollständig geöffnetem Tor ignoriert wird.

```
void main()
{
    ...

    char cGarageOeffnen;
    char cTorIstOben;
    char cRotAn;
    char cGruenAn;

    ...

    while(1) //Hauptschleife
    {
        readInputs(&cGarageOeffnen, &cTorIstOben);

        if( cGarageOeffnen == 1 && cTorIstOben == 0)
        {
            cRotAn = 1;
            cGruenAn = 0;
            ...
        }
        if( cTorIstOben == 1 && cGarageOeffnen == 0)
        {
            cRotAn = 0;
            cGruenAn = 1;
            ...
        }

        writeOutputs(cRotAn, cGruenAn);
    }
}
```

f) Die Funktion `readInputs` schreibt die aktuellen Sensorwerte in die Variablen `cGarageOeffnen` und `cTorIstOben` der main-Funktion aus Teilaufgabe e). Ergänzen Sie die Variablentypen in der Funktionsdefinition.

```
void readInputs( char* pcTorIstOben, char* pcGarageOeffnen)
```

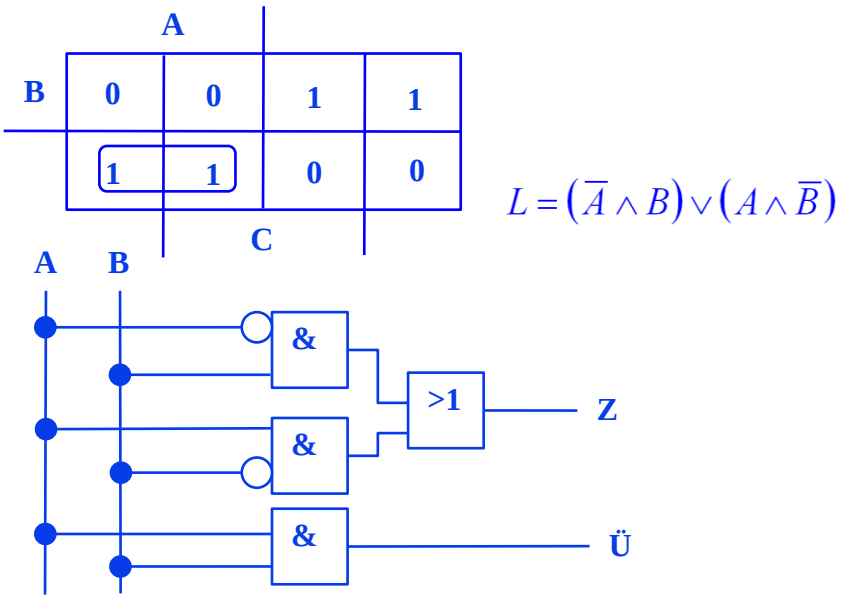
Vorname, Name _____ Matrikelnummer _____

Punkte

g) Gegeben sind die logischen Formeln Z und Ü. Minimieren Sie die Formel Z im *KV-Diagramm* und geben Sie anschließend die logische Schaltung von Z und Ü *in einem Schaltnetz* an.

$Z = (\overline{A} \wedge B \wedge \overline{C}) \vee (\overline{A} \wedge B \wedge C) \vee (A \wedge \overline{B} \wedge \overline{C}) \vee (A \wedge \overline{B} \wedge C)$

$\ddot{U} = (A \wedge B)$



h) Implementieren Sie die gegebene Formel Z aus Aufgabenteil g) in der Programmiersprache C. Verwenden Sie ausschließlich logische Operationen (keine bitweisen Operationen.).

```
void main()
{
    char cA,cB,cC;
    char cZ;

    ...

    cZ = !cA&&cB&&!cC||!cA&&cB&&cC||
         cA&&!cB&&!cC||cA&&!cB&&cC;
}
```

Vorname, Name

Matrikelnummer

Punkte

i) Wie sieht die Ausgabe des folgenden Programms in der Konsole aus?

```
#include <stdio.h>

int main (void)
{
      char cA,cB=2;
      int iC=0;
      int iD=(int)1.7;

      cA=cB*0xA;
      printf("%d %x\n",cA,cB+10);
      iC=0x5+10;
      printf("%o %x\n",iC,iC);
      printf("%d %d",iD,iD+1);

      return 0;
}
```

```
20 c
17 f
1 2
```

Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe BS: Betriebssysteme

Aufgabe BS: 64 Punkte

a) Gegeben ist folgendes Szenario:
Sie sollen für eine Anlagensteuerung ein geeignetes Betriebssystem wählen. Die Anlage muss fünf Prozesse simultan ausführen, die sich drei Betriebsmittel teilen. Die Zeitanforderungen der Anlage bewegen sich im Bereich kleiner 1 ms, da sonst Schäden an den Maschinen und die Verletzung von Menschen nicht ausgeschlossen werden können.
Zur Auswahl stehen Ihnen drei potentielle Betriebssysteme (BS1, BS2, BS3).
Wählen Sie das am besten geeignete und falls vorhanden auch einen Scheduler.
Begründen Sie ihre Wahl (Warum wurde keines der anderen gewählt?)

BS1: Singletasking/Singleuser, echtzeitfähig, keine Semaphorfunktion,
Scheduleroptionen: keine
BS2: Multitasking/Singleuser, nicht echtzeitfähig, verfügt über Semaphorfunktion,
Scheduleroptionen: Round Robin, First in First out, Prioritäten
BS3: Multitasking/Multiuser, echtzeitfähig, verfügt über Semaphorfunktion,
Scheduleroptionen: Round Robin, First in First out, Prioritäten

BS3 mit Prioritäten

Gründe:

MT nötig wegen simultanen Prozessen,

Echtzeitfähigkeit nötig wg Sicherheit+ Scheduler Prioritäten um echtzeitfähig zu sein,

+ Semaphore um Zugriff auf Betriebsmittel zu regeln

Vorname, Name _____ Matrikelnummer _____

Punkte

b) Erklären Sie kurz die Begriffe *asynchrone/synchrone Programmierung, preemptiv und nicht-preemptiv.* Welche Art der Programmierung ist für das *in Aufgabe a) vorgestellte System am besten geeignet* ?
Kreuzen sie in der gegebenen Tabelle an bei welchen Anforderungen (*Echtzeitfähigkeit, Reaktionsfähigkeit auf unvorhergesehene Ereignisse (Flexibilität), Antwortzeiten, Datenkonsistenz)* Sie welche Kombinationen sinnvoll einsetzen können.

Asynchrone Programmierung:

Ereignisgesteuert, bei Systemen mit Flexibilität und ohne Echtzeitanforderungen sinnvoll

Synchrone Programmierung:

Zeitgesteuert, aber unflexibel, bei Systemen mit Echtzeitanforderungen sinnvoll

Preemptiv:

Prozesse höherer Priorität unterbrechen Prozesse niederer Priorität

Nicht-Preemptiv (kooperativ):

Prozesse werden nicht unterbrochen oder nur an sog. Scheduling points.

Für System aus a) am besten geeignet::

Synchrone, preemtive Programmierung, wegen Echtzeitfähigkeit und hoher Antwortzeitanforderungen

Art der Progr. / Anforderung	Async. preemptiv	Sync. preemptiv	Async. nicht preemptiv	Sync. nicht preemptiv
Harte Echtzeit		X		X
Genaue Antwortzeiten	X	X		
Datenkonsistenz			X	X
Flexibilität	X		X	

Vorname, Name _____ Matrikelnummer _____

Punkte

c) Sie wollen die Effizienz und den Durchsatz des Scheduling-Verfahrens Round Robin für eine vorgegebene Prozessanzahl und –dauer bei verschiedenen Parametern (Zeitschlitzgröße, Kontextwechseldauer) bestimmen.
Gegeben sind die folgenden fünf Prozesse:

Prozess	1	2	3	4	5
Dauer	15	23	7	39	55

Das Round-Robin-Verfahren ist in der folgenden Funktion implementiert:

```
void schedRoundRobin( double* pdThroughput, double* pdEfficiency,
int* piProcessList, int iListSize, int iTimeslice, int iContextswitch );
```

Berechnen Sie die Qualitätskriterien (Durchsatz und Effizienz) für Zeitschlitzgrößen von 10 bis einschließlich 100 in 10er Schritten und einer Kontextwechselzeit (iContextswitch) von 1 und geben Sie die Ergebnisse formatiert auf der Konsole aus. Die Qualitätskriterien geben Sie auf zwei Nachkommastellen genau aus.

```
int main()
{
    // Variablendeklaration
    int i;
    double dThroughput = 0.0;
    double dEfficiency = 0.0;
    printf( „Slice\tThroughput\tEfficiency\n“ );
    // Zeitschlitzgrößen durchlaufen
    for( i = 10; i <= 100; i += 10 )
    {
        // Prozessliste definieren
        int piProcessList[ 5 ] = { 15, 23, 7, 39, 55 };
        // Aufruf des Round-Robin-Verfahrens
        schedRoundRobin( &dThroughput, &dEfficiency,
        piProcessList, 5, i, 1 );
        // Ausgabe der Qualitätskriterien auf der Konsole
        printf( „%i\t%.2f\t%.2f\n“, i,
                dThroughput, dEfficiency );
    }
    // Anwendung mit Rückgabewert 0 beenden
    return 0;
}
```

Vorname, Name _____ Matrikelnummer _____

Punkte

d) Gegeben ist die Anordnung von Semaphor-Operationen am *Anfang und am Ende der Tasks A,B,C*. Ermitteln Sie für die Fälle I, II ,III *ob und in welcher Reihenfolge* diese Tasks bei der angegebenen Initialisierung der Semaphor-Variablen ablaufen. Ergibt sich eine *Wiederholungsreihenfolge* und wenn ja geben sie diese an.
Hinweis: Sind mehrere Tasks ablauffähig gilt die Priorität A>B>C. Geben Sie die Reihenfolge der ablaufenden Tasks an z.B. ABCABB an. P(Si) senkt Si um 1 V(Si) erhöht Si um 1.

Task	**A**	**B**	**C**
	P(SA)	P(SB)	P(SC)
	P(SA)		P(SC)
	P(SA)		
	…	…	…
		V(SA)	V(SB)
t	V(SC)	V(SA)	V(SB)

Fall	SA	SB	SC
I	1	1	0
II	4	0	1
III	2	0	1

I. Deadlock nach BA

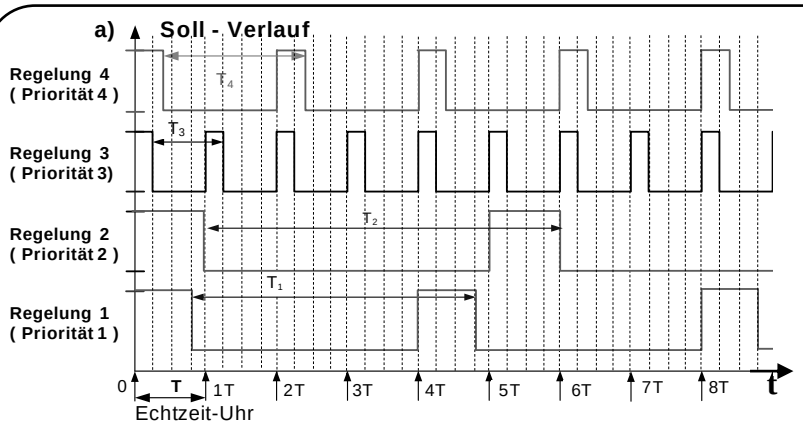
II. Folge ACBAB, Deadlock im Zustand SA2 SB0 SC1

III. Deadlock

Vorname, Name _____ Matrikelnummer _____

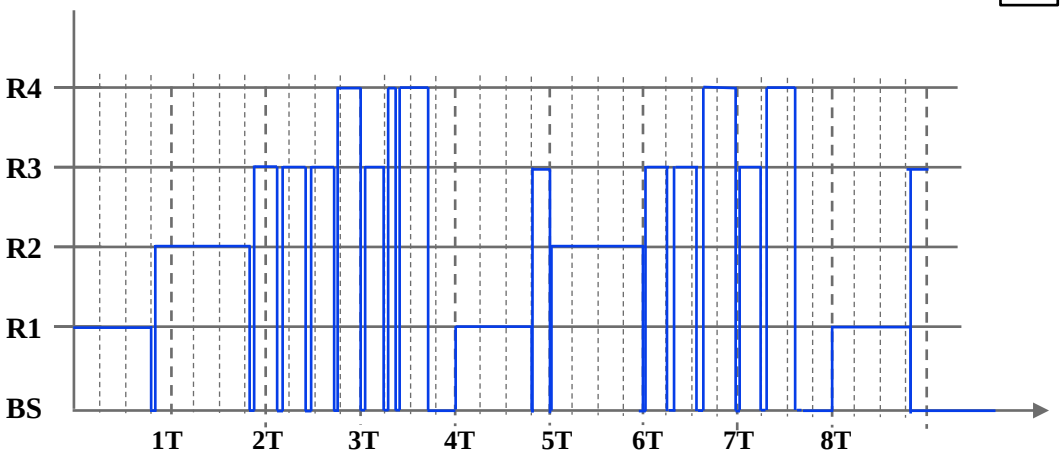
Punkte

e) Stellen Sie das Ist-Systemverhalten der Programmierart *asynchron-preemptiv* in dem angegebenen Diagramm da. Welche Probleme können sich hier zu welchem Zeitpunkt ergeben? Nennen Sie zwei.

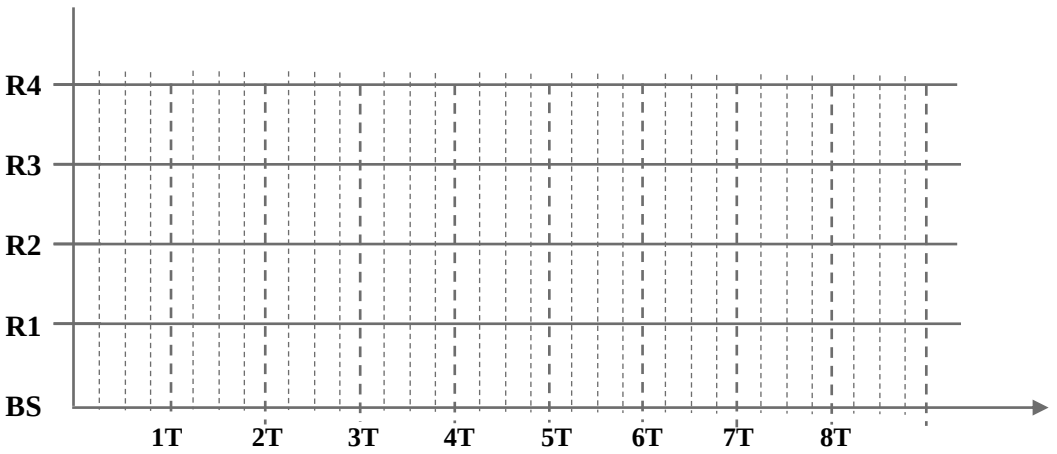


Lösungsfeld 1 und Lösungsfeld 2 können zum Bearbeiten der Aufgabe genutzt werden (Bei Verzeichnen). Markieren Sie welches Feld gewertet werden soll (eines von beiden).

Asynchron-preemptive Programmierung: **Lösungsfeld 1 werten** ☐



Asynchron-preemptive Programmierung: **Reservelösungsfeld werten** ☐



Vorname, Name _____ Matrikelnummer _____

Punkte

f) Sie möchten im Bereich der Embedded-Entwicklung über die RS232-Schnittstelle auf Entwicklerboards zugreifen. Definieren Sie eine Struktur SCHNITTSTELLE _S mit folgenden Elementen: iBaudRate (Typ: Integer), szSender (Array mit 20 Elementen vom Typ Char) und cParity (Typ: Char). Definieren Sie die Struktur gleichzeitig als Typ SCHNITTSTELLE .

```
typedef struct SCHNITTSTELLE_S

{

        int iBaudRate;

        char szSender[20];

        char cParity;

} SCHNITTSTELLE;
```

f) Erzeugen Sie eine Variable mit dem Namen „Testschnittstelle“ vom obigen Typ SCHNITTSTELLE und initialisieren Sie die Elemente durch eine Initialisierungsliste mit folgenden Werten:

- iBaudRate: 10
- szSender: „Testsender“
- cParity: 1

```
SCHNITTSTELLE Testschnittstelle = {10,"Testsender",1};
```

Vorname, Name _____ Matrikelnummer _____

Punkte

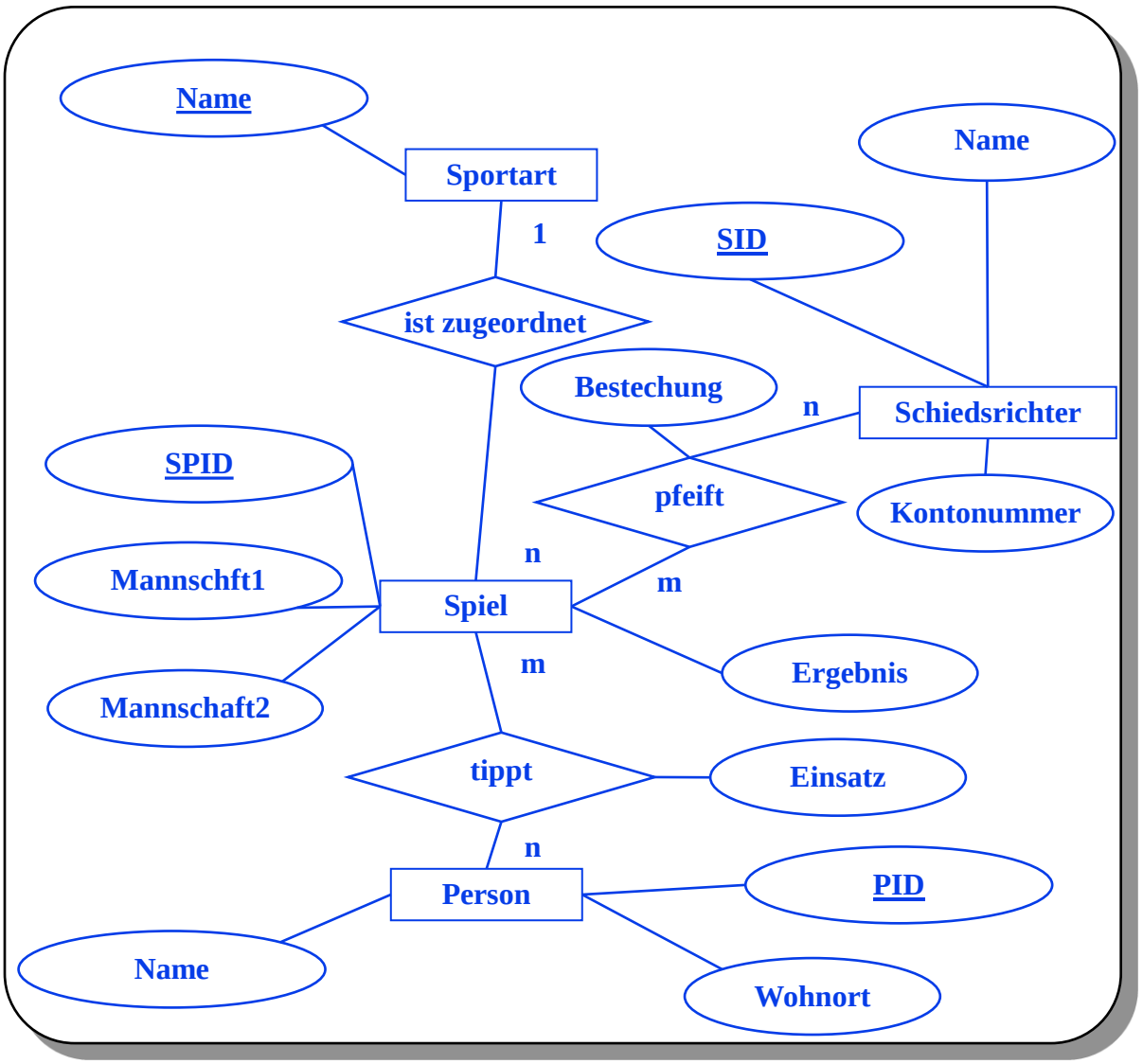
Aufgabe DB: Datenbanken

Aufgabe DB: 30Punkte

a) Herr Paul Krake möchte zur Verwaltung von Wettgeschäften eine Datenbank anlegen. Diese soll es ermöglichen, den Einsatz unterschiedlicher Personen auf unterschiedliche Spiele und ebenso die bezahlten Schiedsrichter zu dokumentieren. Folgende Randbedingungen gelten: Eine *Person* kann auf mehrere *Spiele* mit unterschiedlichen *Einsätzen* tippen. Ein Spiel ist genau einer *Sportart* zugeordnet. Ein Spiel wird von beliebig vielen *Schiedsrichtern* gepfiffen welche mit unterschiedlichen *Summen* bestochen werden können.
Folgende Attribute sind gegeben:

- Sportart: *Name*
- Spiel: *Mannschaft1, Mannschaft2, Ergebnis*
- Person: *Name, Wohnort*
- Schiedsrichter: *Name, Kontonummer*

Erstellen Sie das entsprechende ER-Diagramm. Wählen Sie dabei sinnvolle Schlüsselattribute und ergänzen Sie fehlende Attribute



Vorname, Name _____ Matrikelnummer _____

Punkte

b) Wie viele Tabellen sind *mindestens* notwendig um die in *Aufgabe a* beschrieben Anforderungen in eine relationale Datenbank zu überführen? Begründen Sie ihre Aussage. (*Hinweis: Ohne Begründung keine Punkte*)

6
4 Sind nötig um die Objekte abbilden zu können
Eine für die Relation Schiedsrichter – Spiel
Eine für die Relation Person- Spiel
Die Relation Spiel – Sportart benötigt keine Tabelle, da diese Beziehung in einem zusätzlichen Attribut von Spiel darstellbar ist.

Wichtig:
Für alle weiteren Teilaufgaben ist folgender unvollständiger Datenbankausschnitt gegeben.
Die Tabelle Sportart wurde für die folgenden Aufgaben um zwei Attribute erweitert.

Schiedsrichter		
SID	**Name**	**Kontonummer**
1039	Fritz Müller	1056100
1040	Max Mustermann	1596515
1041	Hans Maier	1235182

Spiel			
SPID	**Mannschaft1**	**Mannschaft2**	**Ergebnis**
1	Spanien	Deutschland	1:0
2	Serbien	Deutschland	1:0
3	Ghana	Uruguay	3:5

Pfeift		
SID	**SPID**	**Bestechung**
1039	1	300.000
1041	1	250.000
1040	2	70.000

tippt	
SPID	**PID**
1	02
1	04
2	03

Person		
PID	**Name**	**Wohnort**
02	Peter Maier	Garching
03	Hansen Müller	Erding
04	Sepp Maier	Dasing

Sportart		
Name	**Ballgröße**	**Spielzeit**

Vorname, Name _____ Matrikelnummer _____

Punkte

c) Legen Sie mit einem SQL Befehl die Tabelle Sportart an. Befüllen Sie die Tabelle Sportart anschließend mit einem SQL Befehl mit folgenden Werten: Name(Fußball); Ballgröße(68,8); Spielzeit(90).

```
CREATE TABLE Sportart (Name VARCHAR(25) PRIMARY KEY,
                       Ballgröße FLOAT,
                       Spielzeit INT);
```

```
INSERT INTO Sportart (Name, Ballgröße, Spielzeit)
VALUES („Fußball“, „68,8“, „90“);
```

d) Geben Sie den SQL Befehl an, mit dem Sie sich die *Namen* der *Personen* anzeigen lassen, die auf ein Spiel getippt haben, welches der Schiedsrichter *„Fritz Müller“* pfeift. Welche Antwort belkommen Sie bei den gegebenen Tabellen?

```
SELECT Name FROM Personen WHERE PID =
SELECT PID FROM tippt WHERE SPID =
SELECT SPID FROM pfeift WHERE SID =
SELECT SID FROM Schiedsrichter WHERE Name = „Fritz Müller“;
```

Peter Maier
Sepp Maier

Vorname, Name

Matrikelnummer

Punkte

Aufgabe MO: Modellierung

Aufgabe MO: 50 Punkte

a) Definieren Sie kurz die Begriffe „Klasse“ und „Objekt“ der UML (Unified Modeling Language) und beschreiben Sie kurz in einem separaten Punkt den Unterschied zwischen beiden.

Objekt: Gegenstand, Person, Begriff mit eindeutiger, nicht veränderbarer Identität (Objektreferenz) und charakteristischen Eigenschaften (Attributwerte),

Klasse: Struktur aus Eigenschaften und Funktionalität

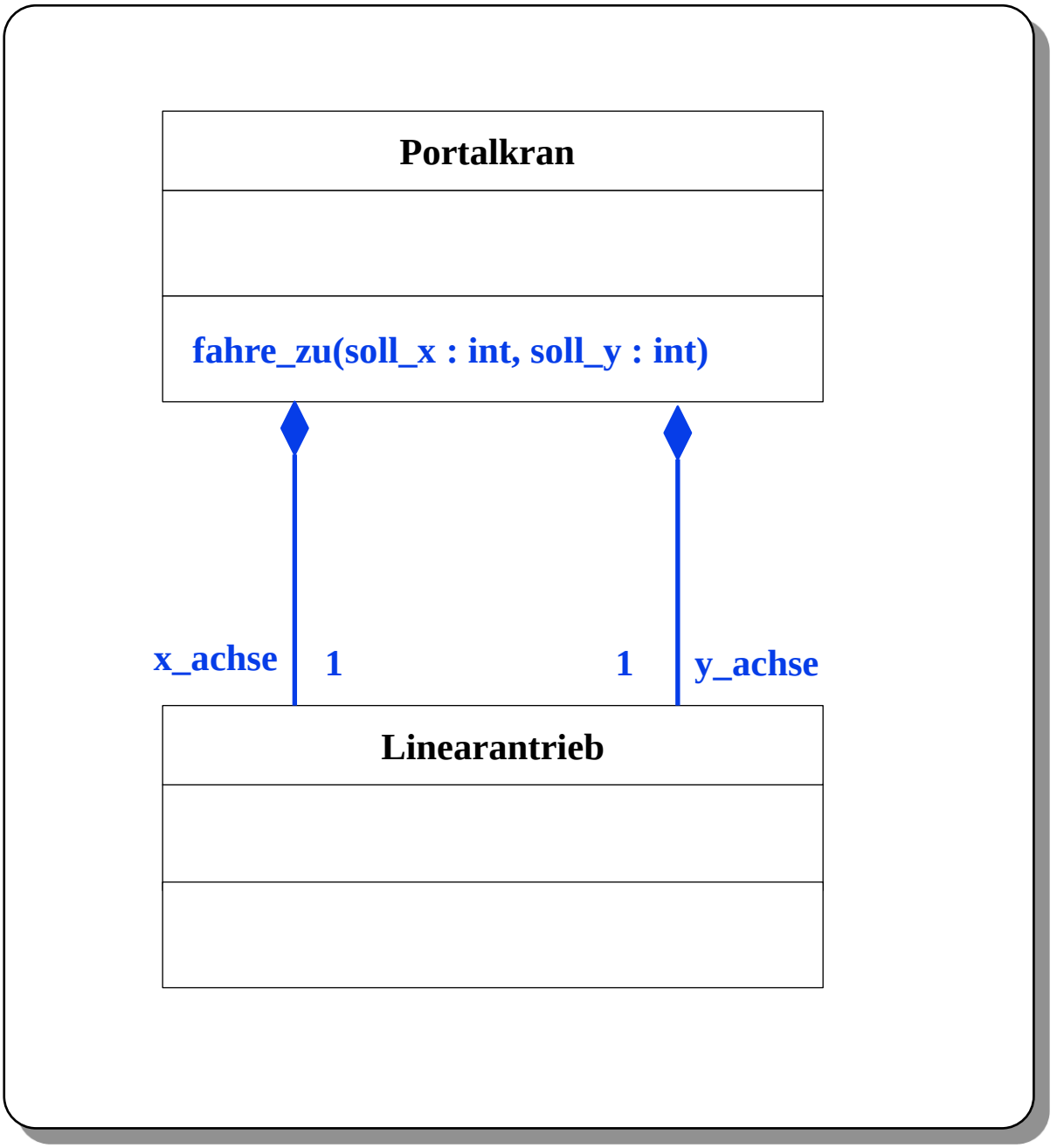
Klasse ist Instanzvorlage einer Menge von Objekten
oder:
Objekt ist Ausprägung (Instanz) einer Klasse

Vorname, Name _____ Matrikelnummer _____

Punkte

b) Gegeben sei das Klassendiagramm eines Portalkrans in UML. Vervollständigen Sie das Diagramm im Lösungsfeld um die Modellierung der folgenden Aspekte:

- Für die Bewegung in x-Richtung und in y-Richtung enthält der Portalkran zwei Instanzen des Linearantriebs als *existenzabhängige* Teile. Die eine Instanz soll über den Instanznamen (Rollennamen) „x_achse“ und die andere über den Instanznamen (Rollennamen) „y_achse“ angesprochen werden können.
- Der Portalkran verfügt über eine Operation, mit der eine definierte Sollposition angefahren werden kann. Die Sollposition wird der Operation *durch je eine ganzzahlige (Integer) Eingangsvariable* für die Sollposition in x-Richtung und für die Sollposition in y-Richtung übergeben.



Vorname, Name

Matrikelnummer

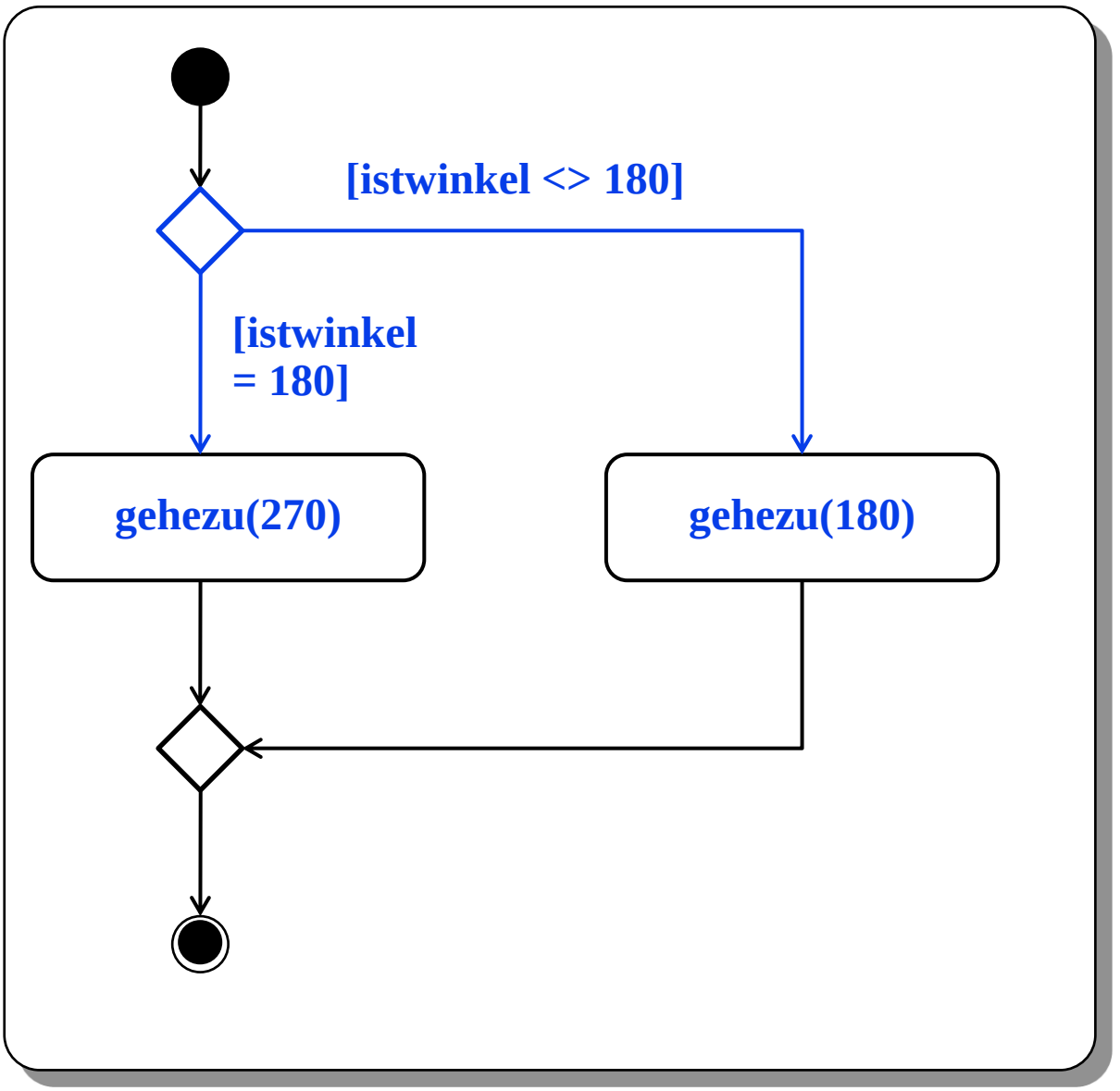
Punkte

c) Gegeben sei eine Klasse zur Beschreibung einer Weiche eines Transportsystems:

Weiche
+ istwinkel : int
gehezu(sollwinkel : int)

Vervollständigen Sie unter Verwendung der gegebenen Attribute und Operationen dieser Klasse das unten im Lösungsfeld stehende Aktivitätsdiagramm, sodass damit folgendes Verhalten beschrieben wird:

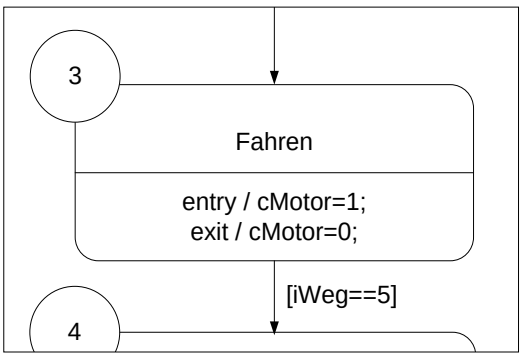
Befindet sich die Weiche nicht in der Stellung 180° so soll sie in diese Stellung fahren. Andernfalls soll sie in die Stellung 270° fahren. Hat sie eine der beiden Stellungen erreicht, ist die Aktivität beendet.



Vorname, Name _____ Matrikelnummer _____

Punkte

d) Vervollständigen Sie die Implementierung des rechts dargestellten Zustands (Ausschnitt aus einem Zustandsdiagramm der UML) in der Programmiersprache C.




```
switch (iStateID)
{
    ...

    case 3:
        if  (__iWeg == 5___)
        {
            __cMotor = 0;_____________
            iStateinitial=1;
            StateID=4;
            break;

        }
        if  (iStateinitial==1)
        {
            __cMotor = 1;_____________
            iStateinitial=0;
        }
        break;

        ...
}
```

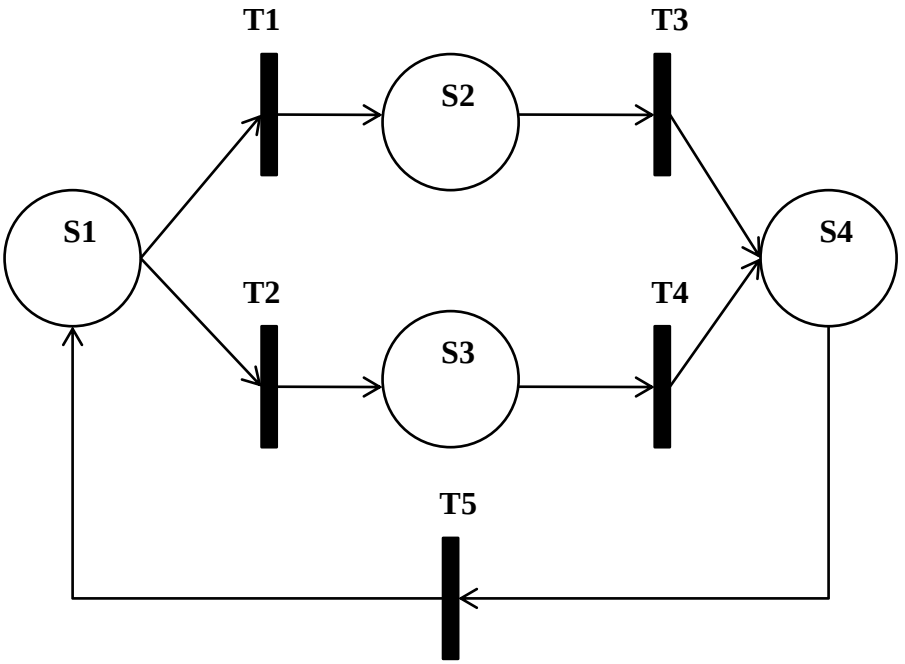
e) Falls in dem Programm in Aufgabenteil d) die Anweisungen des ersten if-Blocks ausgeführt werden (u.a. `iStateinitial=1`), wird dann grundsätzlich auch der zweite if-Block ausgeführt? Begründen Sie Ihre Antwort.

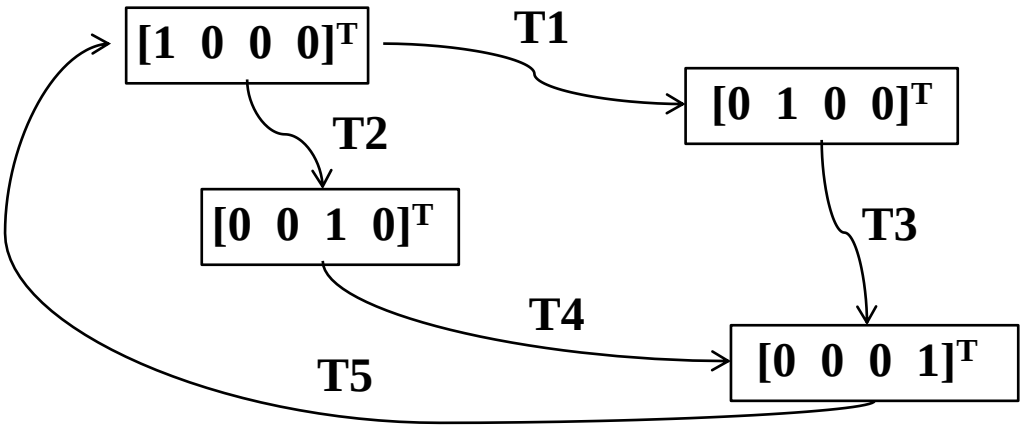
Nein, da durch **`break`** das Switch-Case verlassen wird

Vorname, Name _____ Matrikelnummer _____

Punkte

Wichtig:
Für alle weiteren Teilaufgaben ist folgendes Petrinetz mit der Anfangsmarkierung $M_0 = [1\ \ 0\ \ 0\ \ 0]^T$ sowie der Erreichbarkeitsgraph gegeben.



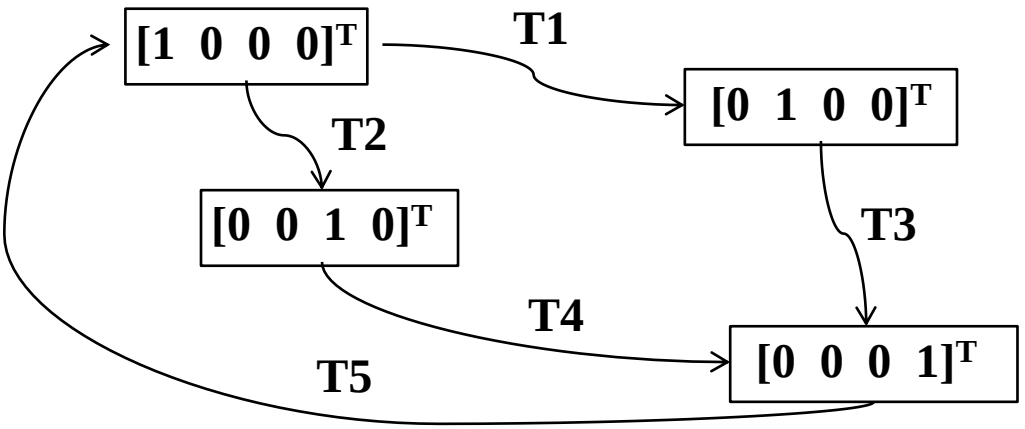


f) Ist das gegebene Petrinetz reversibel? Begründen Sie kurz Ihre Antwort.

Ja, denn der Anfangszustand kann aus jedem folgenden Zustand wieder erreicht werden.

Vorname, Name _____ Matrikelnummer _____

Punkte



g) Stellen Sie die Inzidenzmatrix für das Petrinetz auf.

```
-1 -1  0  0  1
 1  0 -1  0  0
 0  1  0 -1  0
 0  0  1  1 -1
```

h) Ergänzen Sie eine Initialisierungsliste für das Array `iAInzidenzmatrix` mit den Werten aus Teilaufgabe g)

```
int iAInzidenzmatrix[5][4]= {{-1,1,0,0},{-1,0,1,0},
 {0,-1,0,1},{0,0,-1,1},{1,0,0,1}}                    ;
```

i) Vervollständigen Sie die Implementierung des Schaltvorgangs der Transition T3. Die Variable `iAMarkierung` stellt die Markierung des Petrinetzes dar, die neue Markierung (nach dem Schaltvorgang) soll auch wieder dort gespeichert werden.

```
int iAInzidenzmatrix[5][4]= ... ;
int iAMarkierung[4]={0,1,0,0};
int i;
for (i=0;i<4;i++)
{
    iAMarkierung[i] += iAInzidenzmatrix[2][i]          ;
}
```

Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe RK: Rechnerkommunikation

Aufgabe RK: 46 Punkte

a) Zwischen Bedienrechner und Server wird ein binär codiertes Datenpaket ausgetauscht. Das 8 Byte lange Nutzdatenpaket, wie es der Bedienrechner versendet, sowie das serverseitig empfangene Paket sind dargestellt.

- Hat der Server ein *korrektes Paket empfangen*?
- Können Fehler generell mittels dem *zweidimensionalen Paritätsprüfverfahren* erkannt werden?
- Berechnen Sie die dazu notwendigen Prüfsummenwerte *(even-parity)*. Kann der Server das Paket *vollständig korrigieren*? Wenn nein, warum nicht?

Gesendetes Paket

1	0	1	1	1	0	1	0	**1**
0	0	1	0	1	0	0	1	**1**
1	1	1	0	0	0	1	1	**1**
1	0	1	0	1	0	1	0	**0**
0	0	0	1	1	1	1	1	**1**
0	1	0	1	0	1	0	1	**0**
1	1	0	1	0	1	0	0	**0**
0	1	1	0	1	0	1	1	**1**
0	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**1**

Empfangenes Paket

1	0	1	1	1	0	1	0	**1**
0	0	1	1	1	0	0	1	**0**
1	1	1	0	0	0	1	1	**1**
1	0	1	0	1	0	1	0	**0**
0	0	0	1	1	1	1	1	**1**
0	1	0	1	0	1	0	1	**0**
1	1	0	0	0	1	0	0	**1**
0	0	1	0	1	0	1	1	**0**
0	**1**	**1**	**0**	**1**	**1**	**1**	**1**	**1**

- **Das Paket wird fehlerfrei übertragen (Ja / Nein):** **Nein**
- **Das 2D-Paritätsprüfverfahren kann Fehler entdecken (Ja / Nein):** **Ja**
- **Das fehlerhafte Empfangspaket ist eindeutig rekonstruierbar (Ja / Nein):** **Nein**
- **Warum?**

Es kann nur 1 Fehler exakt korrigiert werden, es sind aber 3 Fehler und zwei davon überlagern sich

b) Berechnen Sie die *Hamming-Distanz* zwischen den *ersten beiden Nutzdaten-Bytes* des in Teilaufgabe c) gezeigten, vom Bedienrechner *gesendeten* Datenpakets (ohne die Parity-Bits).

Byte1 =	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**
Byte2 =	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**1**
XOR	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**1**

Hamming-Distanz = 4

Vorname, Name _____ Matrikelnummer _____

Punkte

c)Ein Kistenlager, eine Füllstation und eine Packstation wollen zeitgleich über einen alternativ eingebauten CAN-Bus senden. Sie leiten ihren Sendewunsch durch das Versenden der nachfolgenden hexagonalen Identifier ein:

- Kistenlager-Identifier: 105(hex)
- Füllstation-Identifier: 114(hex)
- Packstation-Identifier: 127(hex)

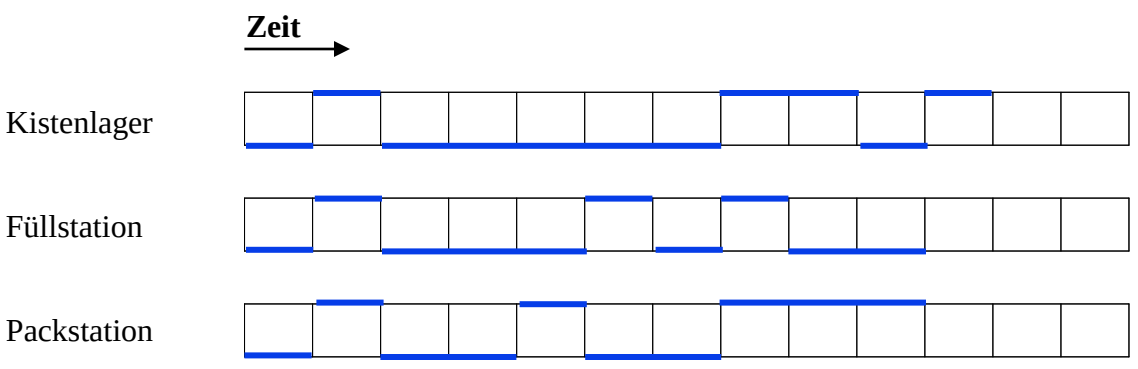
Wie sehen die Identifier binär aus? Geben Sie auch an, ob *Bit-Stuffing* stattfindet, und zudem wenn dies der Fall ist die binäre Zeichenfolge.

105(hex) = **01 0000 0101(binär)**
Mit Bit-Stuffing: **010 0000 1101**

114(hex) = **01 0001 0100(binär)**
Mit Bit-Stuffing: **keine Änderung**

127(hex) = **01 0010 0111(binär)**
Mit Bit-Stuffing: **keine Änderung**

d) Stellen sie die Arbitrierung zur Teilaufgabe c) in einem *Zeit-Diagramm* dar. *Welche Station darf senden*, wenn der High-Pegel rezessiv ist? Wann wird bei welcher Station das Senden abgebrochen? Markieren Sie die Stellen.



Vorname, Name _____ Matrikelnummer _____

Punkte

e) Gegeben sei der nachfolgende Ausschnitt eines C-Programms. Auf den gegebenen Nutzdaten werden verschiedene Operationen ausgeführt. Geben Sie die Werte aller Array-Elemente (iANutzdaten) an, die nach der Ausführung der Anweisungen vorliegen.

```
...
int iANutzdaten[6] = {13,50,39,14,66};
int *piZeiger1 = NULL;
int *piZeiger2 = NULL;

piZeiger1 = &iANutzdaten[2]-1;
piZeiger2 = piZeiger1-1;

*piZeiger2 = *(piZeiger1+=2) * 3;
*(++piZeiger1) = *iANutzdaten%10;

*(++piZeiger2) = *piZeiger2 % 2;
*(piZeiger2) = 'T' - 'S';

*(iANutzdaten+2) = iANutzdaten[0] + 2 % 3;
iANutzdaten[0] == 2;
...
```

iAFeld[0] = 42 iAFeld[1] = 1

iAFeld[2] = 44 iAFeld[3] = 14

iAFeld[4] = 2 iAFeld[5] = undefiniert

Vorname, Name _____ Matrikelnummer _____

Punkte

f) Überführen Sie das folgende Flussdiagramm im angegebenen Lösungsfeld in ein Nassi-Shneiderman Diagramm.