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

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Wintersemester 2010/2011

18. März 2011

Bitte legen Sie Ihren Lichtbildausweis bereit.

Sie haben für die Bearbeitung der Klausur 120 Minuten Zeit.

Diese Prüfung enthält **22** 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	MO	BS	RK	DB	∑	**Note:**
erreichte Punkte							
erzielbare Punkte	**64**	**46**	**60**	**50**	**30**	**244**	

Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe ZS: Zahlensysteme und logische Schaltungen

Aufgabe ZS: 60 Punkte

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

1 (101011)$_2$ (**1121**)$_3$ (**43**)$_{10}$

2 (18)$_{10}$ (**24**)$_7$ (**102**)$_4$

b) Stellen Sie die Gleitkommazahl **-12,043** nach der unten angegebenen Bitstruktur (an IEEE angelehnt) dar und diskutieren Sie die Problematik, die bei der Umrechnung entsteht und wie man dieser entgegenwirken kann.

1	**1010**	**100000**
V	**biased Exponent e (4 Bits)**	**Mantisse (6 Bits)**

12 =(1100)$_2$; Hornerschema für 0,043

0,043*2 = 0,086	**0**	**höchstw. Bit**
0,086*2 = 0,172	**0**	
0,172*2 = 0,344	**0**	
0,344*2 = 0,688	**0**	
0,688*2 = 1,376	**1**	
0,376*2 = 0,752	**0**	

Weiterrechnen nicht sinnvoll da Rechengenauigkeit auf 6 Bit begrenzt!

$$B = 2^{(x-1)} - 1 = 2^{(4-1)} - 1 = 7$$

$$(12{,}043)_{10} \approx (1100{,}00001\ldots)_2$$

$$1{,}100000\ldots 2^3 \Rightarrow E = 3$$

$$e = E + B = 3 + 7 = (10)_{10} = (1010)_2$$

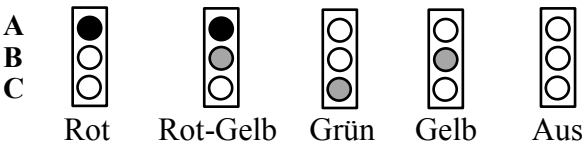
Problem: Durch die Einschränkung der Bits geht die Genauigkeit verloren. Wird wieder zurückgerechnet entsteht die Gleitkommazahl 12 (0,043 geht verloren)!
Wenn nach IEEE 754 Single kodiert wird kann die Zahl mit deutlich höherer Genauigkeit dargestellt werden.

Vorname, Name

Matrikelnummer

Punkte

Aufgabe ZS: Zahlensysteme und logische Schaltungen



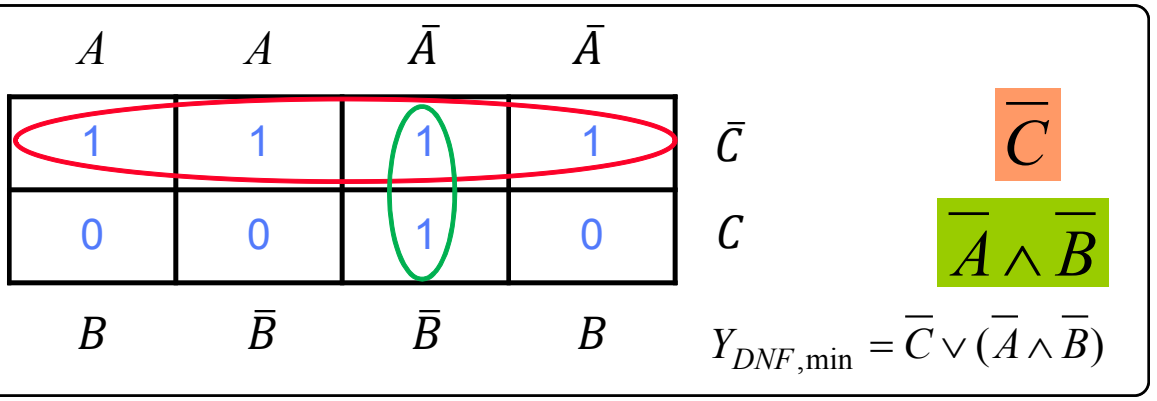
c) Angegeben sind die korrekten möglichen Zustände einer Ampel. Im folgenden soll nun eine Überwachungsschaltung realisiert werden, die bewertet ob die Kombination aus Zuständen (leuchtende rote Birne entspricht z.B. A=1) korrekt und zulässig ist. Vervollständigen Sie hierfür zunächst die unten angegebene Wahrheitstabelle, in der jeder der fünf dargestellten Zustände mit dem Ausgang Y=1 (Ja) bewertet wird. Stellen Sie mit Hilfe der Minterme die Disjunktive Normalform (DNF) anhand der Wahrheitstabelle auf.

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

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

d) Um den Watchdog möglichst kostengünstig umzusetzen, sollen Sie mit Hilfe eines KV-Diagramms die Minimalrealisierung für die DNF aus Aufgabe c) berechnet werden. Falls Sie das Ergebnis aus Aufgabe c) nicht berechnet haben, rekonstruieren Sie das KV-Diagramm mit Hilfe folgender KNF:

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



$Y_{DNF,\min} = \bar{C} \vee (\bar{A} \wedge \bar{B})$

Vorname, Name _____ Matrikelnummer _____

Punkte

e) Nun soll ein kurzes C-Programm vervollständigt werden, welches den Status der Ampel erkennt und überwachen soll. Die Darstellung ist auf die Zustände ROT, ROTGELB (Umschalten), GRÜN und FEHLER beschränkt.
Die restlichen Zustände (GELB, AUS) existieren, sollen aber hier nicht betrachtet werden. Schreiben Sie die benötigten IF-Bedingungen um den Zustand der Ampel eindeutig zu erkennen, sowie die zugehörigen Anweisungen um der Variable *Astatus* den richtigen Wert zuzuweisen.

```
typedef enum {ROT,ROTGELB,GRUEN,GELB,AUS,FEHLER}
AMPEL;
void main()
{
  ...
  char cRotAn;
  char cGruenAn;
  char cGelbAn;
  AMPEL Astatus;
  ...
   while(1) //Hauptschleife
    {
      readInputs(&cRotAn, &cGelbAn, &cGruenAn);
       \\ Erkennung von Schaltung ROT
      if(cRotAn == 1 && cGelbAn == 0 && cGruenAn ==0 )
       \\ Status setzen
      {Astatus=ROT;                              }
       \\ Erkennung von Schaltung ROTGELB
      if(    cRotAn == 1 && cGelbAn == 1 && cGruenAn)==0
       \\ Status setzen
      {Astatus=ROTGELB;                          }
       \\ Erkennung von Schaltung GRUEN
      if(    cRotAn == 0 && cGelbAn == 0 && cGruenAn)==1
       \\ Status setzen
      {Astatus=GRUEN;                            }
     ...
      else Astatus=FEHLER     ;
    ...
         writeOutputs(Astatus);
    }
}
```

f) Im Quelltext aus Aufgabe g) wurde in der ersten Zeile mit *typedef enum* und anschließender Definition der Variable *Astatus* vom Typ *AMPEL* ein benutzerdefinierter Datentyp eingeführt. Wie lautet die Ausgabe einer *printf*-Funktion bei folgender Definition von *Astatus*:

```
AMPEL Astatus=GRUEN; printf(„%i“,Astatus);
```

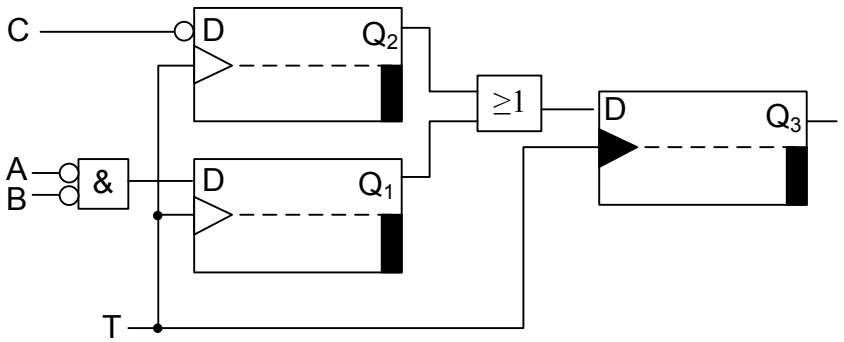
AUSGABE: 2

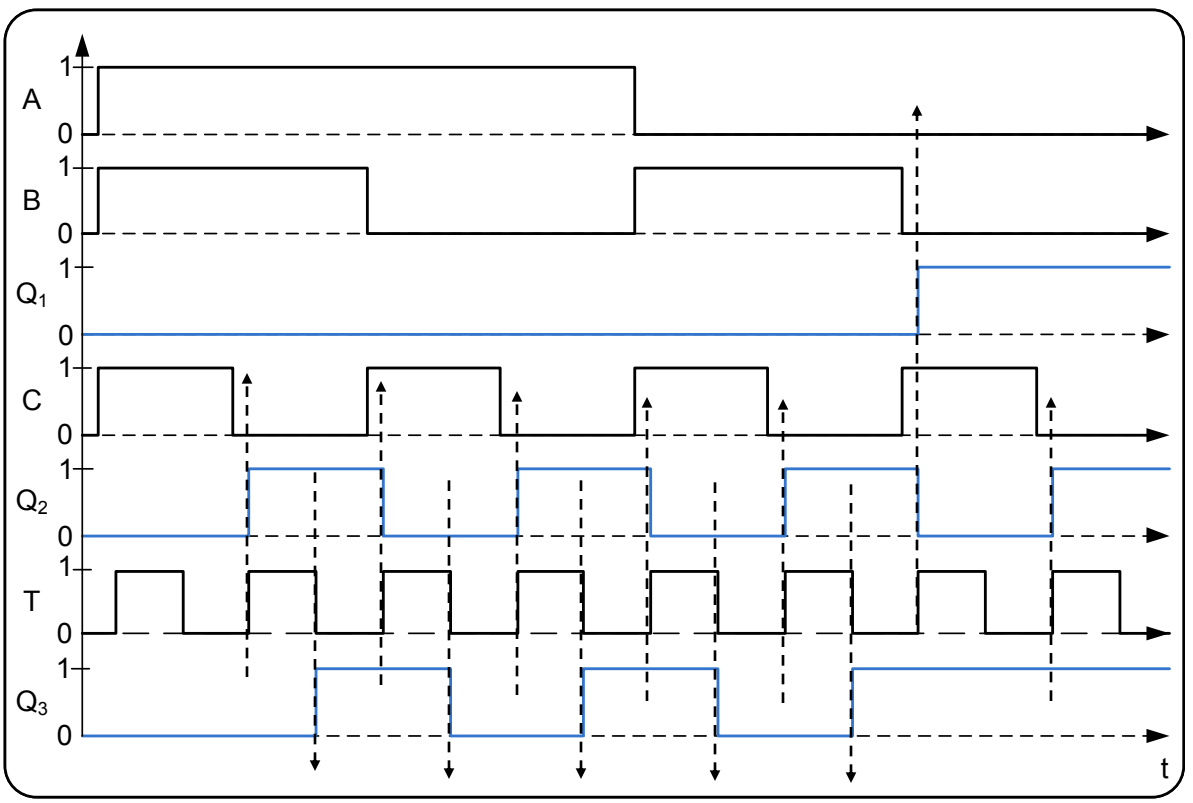
Vorname, Name

Matrikelnummer

Punkte

g) Gegeben ist unten dargestellte FlipFlop-Schaltung, bestehend aus drei Taktflankengesteuerten D-FlipFlops. Erstellen Sie aus dem gegebenen Zeitdiagramm die Ausgänge, Q1, Q2 und Q3 unter Berücksichtigung des Triggersignals T und der drei Eingänge A, B, C. Nehmen Sie als Ausgangssituation an, dass Q1, Q2 und Q3 alle auf Low-Level (Signal am Ausgang ist 0) sind.
Hinweis: Innerhalb eines Q-Diagramms gibt es Punktabzug für einen falschen Ausschlag. Negative Punkte werden nicht in die nächste Q-Zeile mitgenommen.





Vorname, Name _____ Matrikelnummer _____

Punkte

h) Wie sieht die Ausgabe des folgenden C-Programms in der Konsole aus? (Beachten Sie den unten angegebenen erweiterten ASCII-Code bzw. Windows-Standardzeichensatz!)

```
#include <stdio.h>

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

      cA=0xC/cB;
      printf("%d %x\n",cA>>2,17-cB);
      printf("%c %c\n",((++iC&&iD)>=1)+2,85);
      iC=0xB+2;
      printf("%o %x\n",iC,iC);
      system("PAUSE");
      return 0;
}
```

1 f/F/0XF

♥ U

15 d/D/0xD

(mehrere Hex-Darstellungen richtig)

0		24	↑	48	0	72	H	96	`	120	x	144	É	168	¿	192	└	216	╪	240	≡
1	☺	25	↓	49	1	73	I	97	a	121	y	145	æ	169	⌐	193	┴	217	┘	241	±
2	☻	26	→	50	2	74	J	98	b	122	z	146	Æ	170	¬	194	┬	218	┌	242	≥
3	♥	27	←	51	3	75	K	99	c	123	{	147	ô	171	½	195	├	219	█	243	≤
4	♦	28	∟	52	4	76	L	100	d	124	|	148	ö	172	¼	196	─	220	▄	244	⌠
5	♣	29	↔	53	5	77	M	101	e	125	}	149	ò	173	¡	197	┼	221	▌	245	⌡
6	♠	30	▲	54	6	78	N	102	f	126	~	150	û	174	«	198	╞	222	▐	246	÷
7		31	▼	55	7	79	O	103	g	127	⌂	151	ù	175	»	199	╟	223	▀	247	≈
8		32		56	8	80	P	104	h	128	Ç	152	ÿ	176	▒	200	╚	224	α	248	°
9		33	!	57	9	81	Q	105	i	129	ü	153	Ö	177	▓	201	╔	225	ß	249	∙
10		34	"	58	:	82	R	106	j	130	é	154	Ü	178	▓	202	╩	226	Γ	250	·
11	♂	35	#	59	;	83	S	107	k	131	â	155	¢	179	│	203	╦	227	π	251	√
12	♀	36	$	60	<	84	T	108	l	132	ä	156	£	180	┤	204	╠	228	Σ	252	ⁿ
13		37	%	61	=	85	U	109	m	133	à	157	¥	181	╡	205	═	229	σ	253	²
14	♫	38	&	62	>	86	V	110	n	134	å	158	₧	182	╢	206	╬	230	µ	254	■
15	☼	39	'	63	?	87	W	111	o	135	ç	159	ƒ	183	╖	207	╧	231	τ	255	
16	►	40	(	64	@	88	X	112	p	136	ê	160	á	184	╕	208	╨	232	Φ		
17	◄	41	)	65	A	89	Y	113	q	137	ë	161	í	185	╣	209	╤	233	Θ		
18	↕	42	*	66	B	90	Z	114	r	138	è	162	ó	186	║	210	╥	234	Ω		
19	‼	43	+	67	C	91	[	115	s	139	ï	163	ú	187	╗	211	╙	235	δ		
20	¶	44	,	68	D	92	\	116	t	140	î	164	ñ	188	╝	212	╘	236	∞		
21	§	45	-	69	E	93	]	117	u	141	ì	165	Ñ	189	╜	213	╒	237	ø		
22	▬	46	.	70	F	94	^	118	v	142	Ä	166	ª	190	╛	214	╓	238	ε		
23	↨	47	/	71	G	95	_	119	w	143	Å	167	º	191	┐	215	╫	239	∩		

Vorname, Name

Matrikelnummer

Punkte

Aufgabe 2: MO Modellierung

Aufgabe 2: 36 Punkte

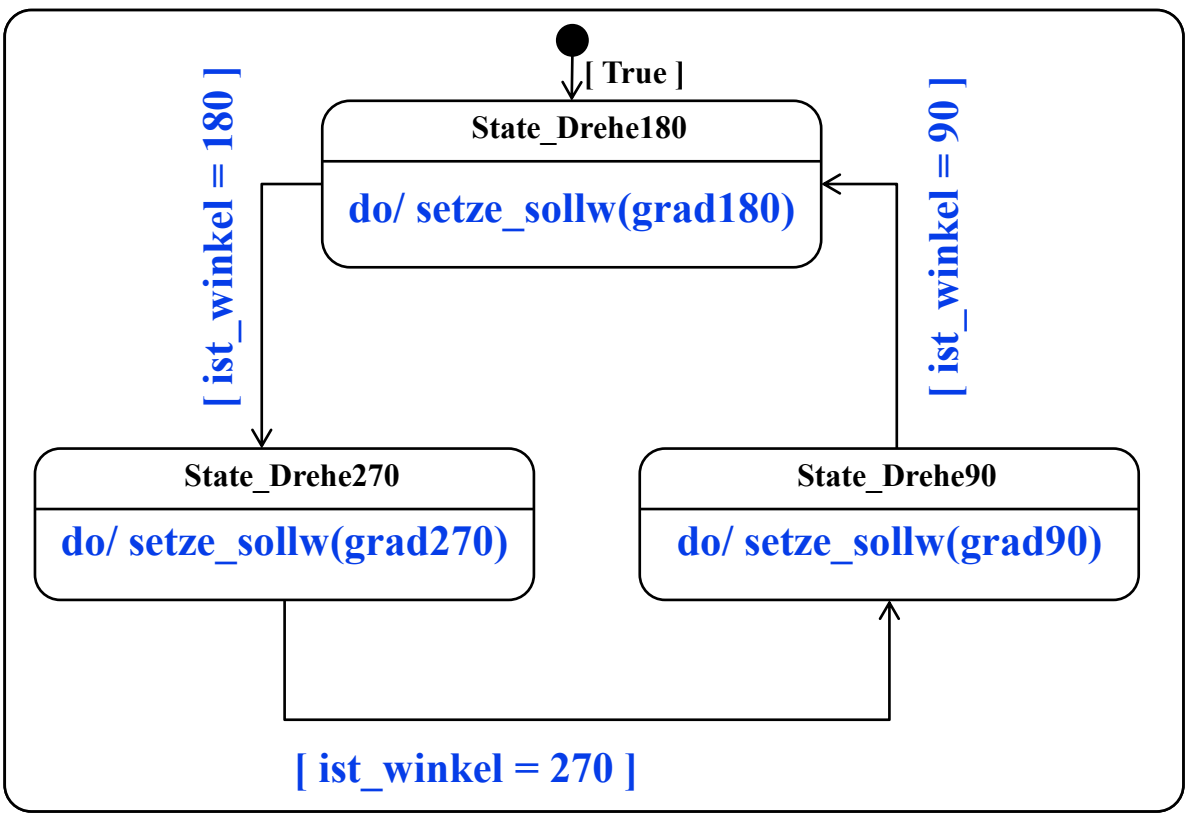
a) Gegeben sei eine Klasse zur Beschreibung einer Weiche eines Transportsystems sowie eine Enumeration. Durch Aufruf der Operation „setze_sollw(in : grad)“ der Weiche kann sie auf einen jeweils beim Aufruf übergebenen Sollwert (grad90 → 90°, grad180 → 180°, grad270 → 270°) gedreht werden.

Weiche
+ istwinkel : int
setze_sollw(in : grad)

<< Enumeration >> grad
grad90 grad180 grad270

Fügen Sie im unten stehenden Zustandsdiagramm Bedingungen für die bereits eingezeichneten Transitionen in Abhängigkeit des Attributs „istwinkel“ ein. Die Zuordnung eines Zustands zum Wert des Attributs ist in der unten stehenden Tabelle angegeben. Fügen Sie darüber hinaus innerhalb der Zustände als „do-Aktivität“ den Aufruf der Operation „setze_sollw(in : grad)“ ein und übergeben Sie dabei den jeweils passenden Sollwert, so dass das Zustandsdiagramm vollständig durchlaufen werden kann. Das Verhalten der Klasse, das mit dem Zustandsdiagramm implementiert werden soll, besteht aus der permanenten Rotation in mathematisch positiver Richtung. Die Weiche soll dieses Verhalten aus dem Zustand „State_Drehe180“ heraus starten.

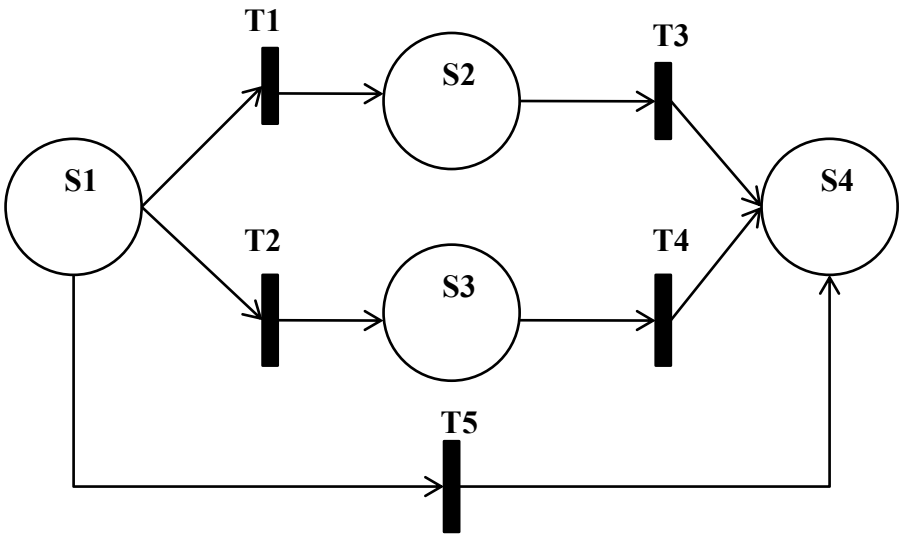
Zustand	Wert „istwinkel“
State_Drehe180	90
State_Drehe270	180
State_Drehe90	270



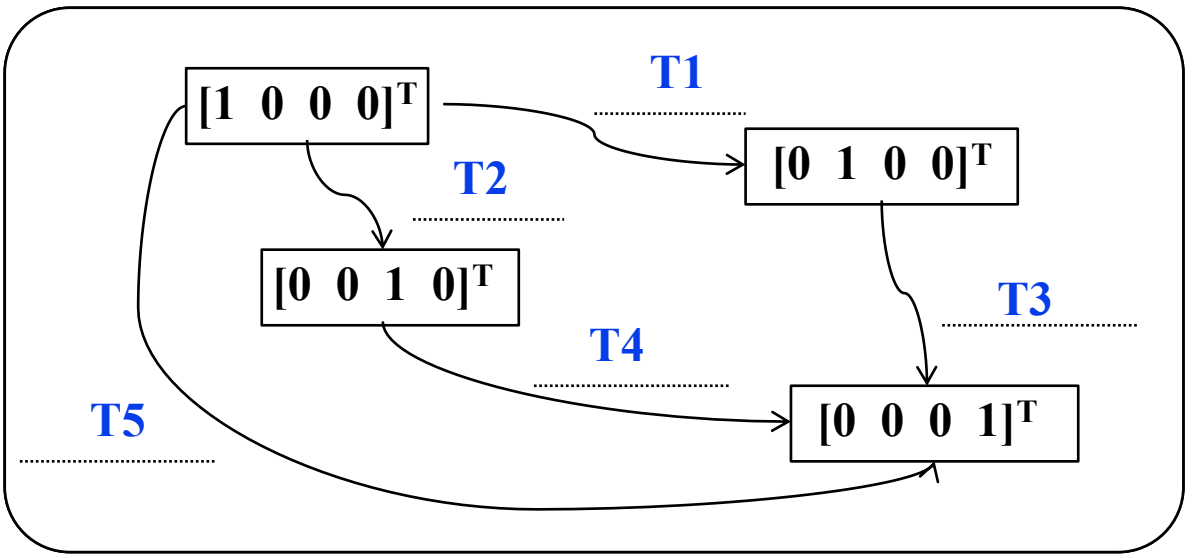
Vorname, Name _____ Matrikelnummer _____

Punkte

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



b) Fügen Sie im Lösungsfeld an den dafür vorgesehen Stellen (gestrichelte Linien) des Erreichbarkeitsgraphs die korrekte Transitionsbezeichnung ein.



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

Nein, denn der Anfangszustand kann nicht wieder erreicht werden.

Vorname, Name _____ Matrikelnummer _____

Punkte

d) Ergänzen Sie die Initialisierungsliste im Lösungsfeld für das Array `iATransitionsvektor5` entsprechend der Transition T5 des auf der vorigen Seite gegebenen Petrinetzes.

```
int iATransitionsvektor5[4]= { -1, 0, 0, 1 } ;
```

e) Vervollständigen Sie die Implementierung des Schaltvorgangs der Transition T5 in der Programmiersprache C. Die Variable `iAMarkierung` stellt die aktuelle Markierung des Petrinetzes dar. Vor dem Schaltvorgang soll in der ersten FOR-Schleife überprüft werden, ob die Werte im Array `iAMarkierung` den Werten im Array `iAStelle1` entsprechen und abhängig davon der Wert der Variable `iSchalten` verändert werden. Der Schaltvorgang soll abhängig vom Wert der Variablen `iSchalten` in der zweiten FOR-Schleife ausgeführt werden. Die neue Markierung (nach dem Schaltvorgang) soll wieder in der Variable `iAMarkierung` gespeichert werden

```
int iATransitionsvektor5[4];
int iAStelle1[4]={1,0,0,0};
int iAMarkierung[4]={1,0,0,0};
int i;
int iSchalten = 1;

/* Erste For-Schleife*/
for (i=0;i<4;i++)
{

    if( iAMarkierung[i] != iAStelle1[i] ) iSchalten = 0 ;
};

if( iSchalten )    // opt.: iSchalten > 0 // opt.: iSchalten == 1
{
     /* Zweite For-Schleife*/
     for(i=0;i<4;i++)
     { iAMarkierung[i] = iAMarkierung[i] + iATransitionsvektor5[i]
       //opt.: iAMarkierung[i] += iATransitionsvektor5[i]
     ;
     };
};
```

Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe BS: Betriebssysteme

Aufgabe BS: 60 Punkte

a) Nennen Sie *stichpunktartig* die drei zentralen Aufgaben eines Betriebssystems.

- Bereitstellung von Hilfsmitteln zur Bearbeitung von Benutzerprogrammen oder „Funktionsbibliothek“
- geregelte Verwaltung und effiziente Nutzung von Betriebsmitteln oder „ordnende Hand"
- Abbilden/Abstraktion der Hardwareschnittstellen auf einheitliches Schema an der Schnittstelle zur Anwendung oder „virtuelle Maschine“

b) Kreuzen Sie in der gegebenen Tabelle an, bei welchen Programmierarten welche Eigenschaften in Zusammenhang mit Betriebssystemeigenschaften vorhanden sind.
Hinweis: Falsche Kreuze führen zu Punktabzug innerhalb der Aufgabe. Minimal können 0 Punkte erreicht werden.

Art der Progr. / Eigenschaften	Asynchron preemptiv	Synchron preemptiv	Asynchron non preemptiv	Synchron non preemptiv
Niederpriore Prozesse können durch höherpriore unterbrochen werden	X	X		
Niederpriore Prozesse können durch höherpriore nicht unterbrochen werden			X	X
Prozessorvergabe zeitgesteuert		X		X
Prozessorvergabe ereignisgesteuert	X		X	

Vorname, Name _____ Matrikelnummer _____

Punkte

c) In der Programmiersprache C soll ein Programm zur Stringverarbeitung implementiert werden. In einem String werden den Buchstaben A, B, C die Werte 0 oder 1 zugeordnet. *Der String entspricht genau dem im Beispiel gezeigten Muster, d.h. es kommen immer genau einmal A, B und C mit ihren Werten in beliebiger Reihenfolge vor und die Wertangaben (z.B. A=1) sind nur durch je ein Semikolon getrennt.*
Beispiel: Die Werte die durch den String `szDS` "A=1;B=0;C=0;" gegeben sind, sollen im Array `caDA [3][2]` entsprechend der unten gezeigten Reihenfolge abgelegt und anschließend ausgegeben werden:

Sich ergebende Reihenfolge in caDA:

caDA =

A	1
B	0
C	0

Ausgabe :

Buchstabe A = 1
Buchstabe B = 0
Buchstabe C = 0

Vervollständigen Sie die Funktion `main()`, die den String `szDS` auswertet und in das Array `caDA` ablegt. Im Anschluss werden die Buchstaben und ihre Werte, wie im Beispiel zeilenweise ausgeben.

```
#include <stdio.h>
void main()
{
   int i=0;
   char caDA [3][2]; //DatenArray
   char* szDS;     //Datenstring

   // Eingabe des Datenstrings nach Muster A=1;B=…
   eingabe(szDS);
   // Analyse des Strings
   while (i< 9-12 )
   {

   switch (szDS[i])

      {  case 'A':  caDA[ 0 ][ 0 ] = 'A';

                    caDA[ 0 ][ 1 ] = szDS[ i+2 ];

                    i= i+4        ;

         break;
           //...case 'B' und case 'C' analog
         default: i++;

      }

   }
for (i=0; i<3; i++)
 printf("Buchstabe %c = %c \n",caDA[ i ][ 0 ],caDA[ i ][ 1 ]);

}
```

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 und III *ob und in welcher Reihenfolge* diese Tasks bei der angegebenen Initialisierung der Semaphor-Variablen ablaufen. Ergibt sich eine Wiederholungsreihenfolge? Wenn ja geben Sie diese an.
Hinweis: Sind mehrere Tasks ablauffähig, gilt die Prioritätenreihenfolge A>B>C, d.h. C wird am nachrangigsten abgearbeitet. Geben Sie die Reihenfolge der ablaufenden Tasks an (z.B. ABCABB). Sobald in einer Prozessreihenfolge ein Fehler vorkommt, werden auf nachfolgende Prozesse keine Punkte mehr vergeben P(Si) senkt Si um 1 V(Si) erhöht Si um 1.

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

Fall	SA	SB	SC
I	0	0	2
II	3	0	0
III	1	0	3

I. Deadlock/Ende nach C,
(da für SA=0<1 SB=1<2 SC=0<2 kein weiterer Prozess ablaufen kann)

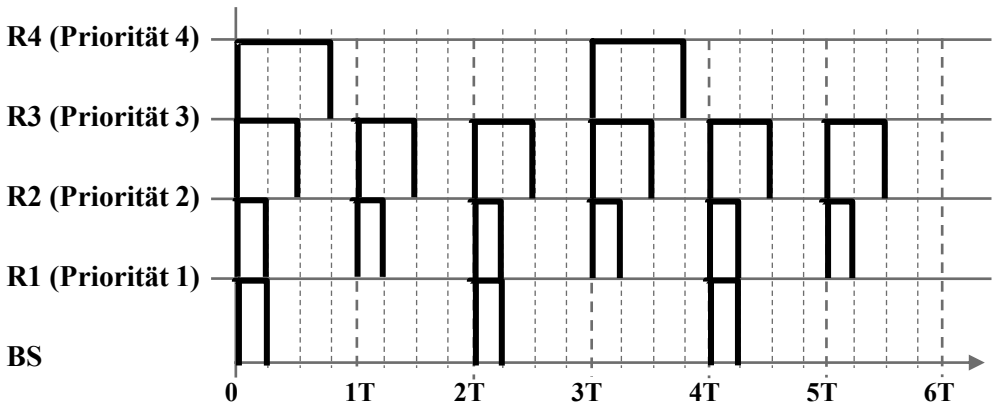
I. Deadlock/Ende nach AAAC

II. Deadlock/Ende nach ACCBAAC

Vorname, Name _____ Matrikelnummer _____

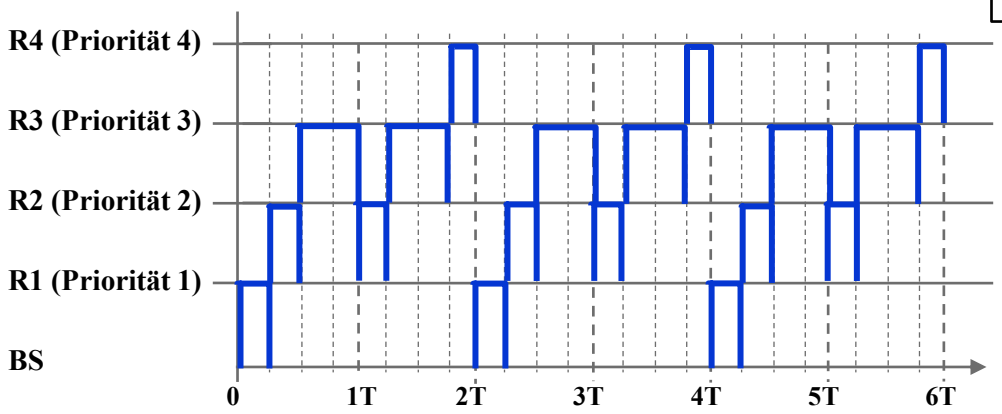
Punkte

e) Stellen Sie das Ist-Systemverhalten der Programmierart *asynchron-preemptiv* in dem angegebenen Diagramm da.
Hinweis: Priorität 1 ist die höchste Priorität (BS>R1>R2>R3>R4).

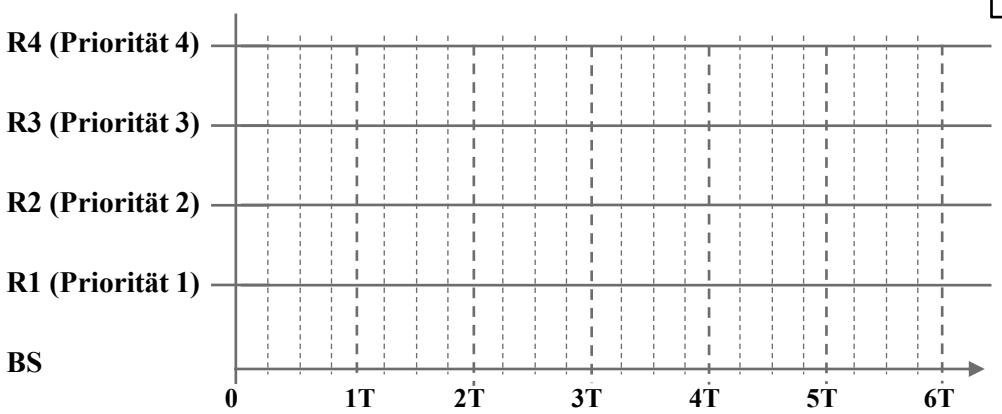


Lösungsfeld 1 und Lösungsfeld 2 können zum Bearbeiten der Aufgabe genutzt werden (Bei Verzeichnen). Markieren Sie welches Feld gewertet werden soll (eines von beiden).

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



Asynchron-preemptive Programmierung: **Reservelösungsfeld werten**



Vorname, Name _____ Matrikelnummer _____

Punkte

f) Sie möchten im Bereich der Scheduler-Entwicklung einen Datentyp für die Prozessverwaltung anlegen. Definieren Sie eine Struktur `PROZESS_S` mit folgenden Elementen: `iProzessUID` (Typ: Integer), `iPrio` (Typ: Integer) und `szName` (Array mit 20 Elementen vom Typ Char). Definieren Sie die Struktur gleichzeitig als Typ `PROZESS`.

```
typedef struct PROZESS_S
{
        int iProzessUID;
        int iPrio;
        char szName[20];
        } PROZESS;
```

g) Gegeben sei eine zweidimensionale Matrix mit den rechts angegebenen Werten.
Bilden Sie diese Matrix als zweidimensionales Array `Zahlenmatrix` in C ab und initialisieren Sie es mit den abgebildeten Werten mittels einer Initialisierungsliste. *Hinweis: Pro Kommafehler wird ein Punkt abgezogen.*

6	8	5
5	8	1
22	71	8

```
/* Variablendeklarartion mit Wertzuweisung */
int Zahlenmatrix[3][3]={{6,8,5},{5,8,1},{22,71,8}};
Alternativ:
Char Zahlenmatrix …
```

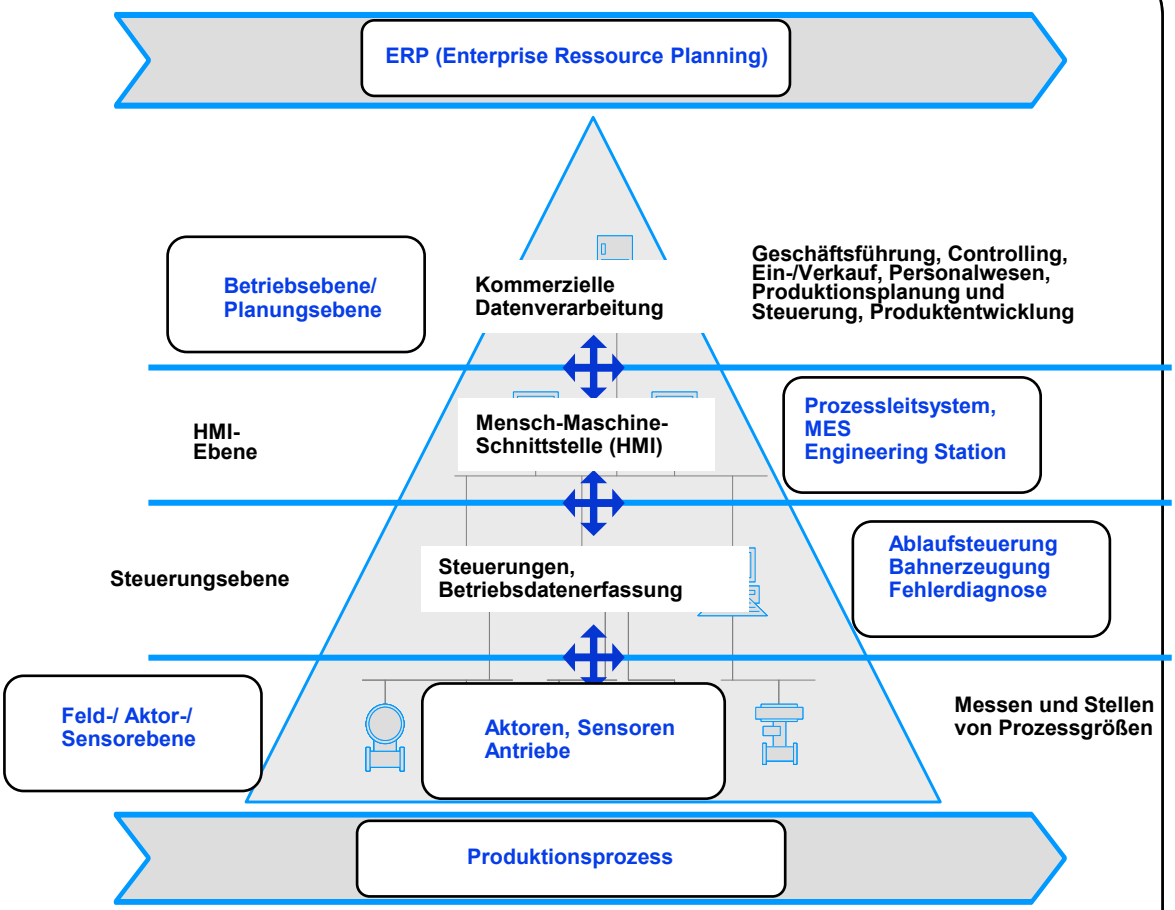
Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe 4: RK Rechnerkommunikation

Aufgabe RK: 50Punkte

a) Vervollständigen Sie die aus der Vorlesung bekannte Abbildung, welche die Informationspyramide der Automatisierungstechnik darstellt, um die fehlenden Begriffe.



b) Berechnen Sie die *Hamming-Distanz* der angegebenen Bytes.

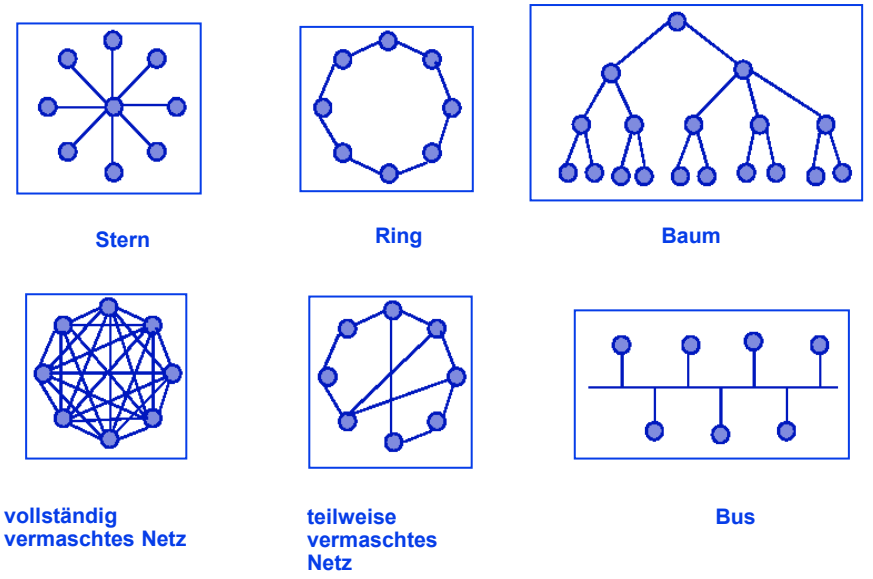
Byte1 =	**1**	**0**	**1**	**1**	**1**	**0**	**0**	**1**
Byte2 =	**0**	**0**	**1**	**0**	**1**	**0**	**1**	**1**
XOR	**1**	**0**	**0**	**1**	**0**	**0**	**1**	**0**

Hamming-Distanz = **3**

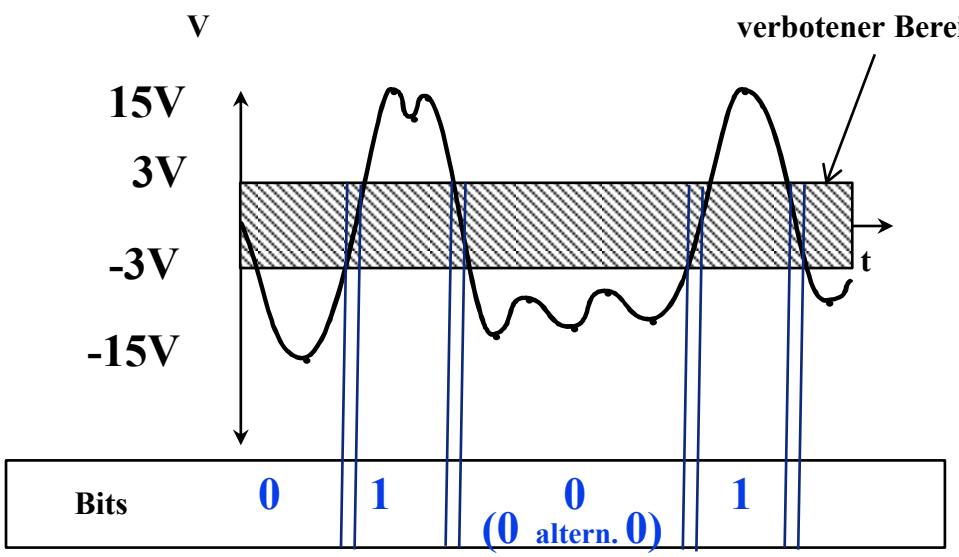
Vorname, Name _____ Matrikelnummer _____

Punkte

c) Benennen uns skizzieren Sie 3 mögliche Bustopologien.



d) Tragen Sie an der folgenden Skizze für das Signal einer RS232 Kommunikation die Signaldurchgänge durch den verbotenen Bereich ein. Ergänzen Sie weiterhin im Feld Bits die entsprechenden Bitwerte für das Signal.



Vorname, Name _____ Matrikelnummer _____

Punkte

e) Zur Berechnung der Querparität eines C-Strings soll die C-Funktion `getParityBit()` implementiert werden. `getParityBit()` gibt 1 zurück falls die Summe der einzelnen Character-Zeichen ungerade ist und 0 falls die Summe gerade ist (siehe Beispiel). Dabei läuft die Funktion die Zeichenkette von vorne zeichenweise durch und berechnet in jedem Schritt das aktuelle Paritätsbit. Gestoppt wird die Schleife erst beim Ende des Strings.

Beispiel:

Zeichen:	**I**	**T**	**M**	→	**Summe**	→	**Querparität**
Wert:	**73**	**84**	**77**		**234**		**0**

```
int getParityBit(char* cTelegram)
{
        // Laufvariable für die Schleife
        int iOffset = 0;
        // Paritätsbit
        int iParity = 0;
        // aktuelles Zeichen in der Zeichenkette
        char cToken = cTelegram[0] ;          (oder cTelegram[iOffset])

        while ( cToken != '\0' ){             (oder cToken != 0)
                iParity = (iParity + cToken) % 2 ;
                iOffset++;
                cToken = *(cTelegram + iOffset) ;   oder cTelegram[iOffset];
        }
        return iParity ;
}
```

Vorname, Name _____ Matrikelnummer _____

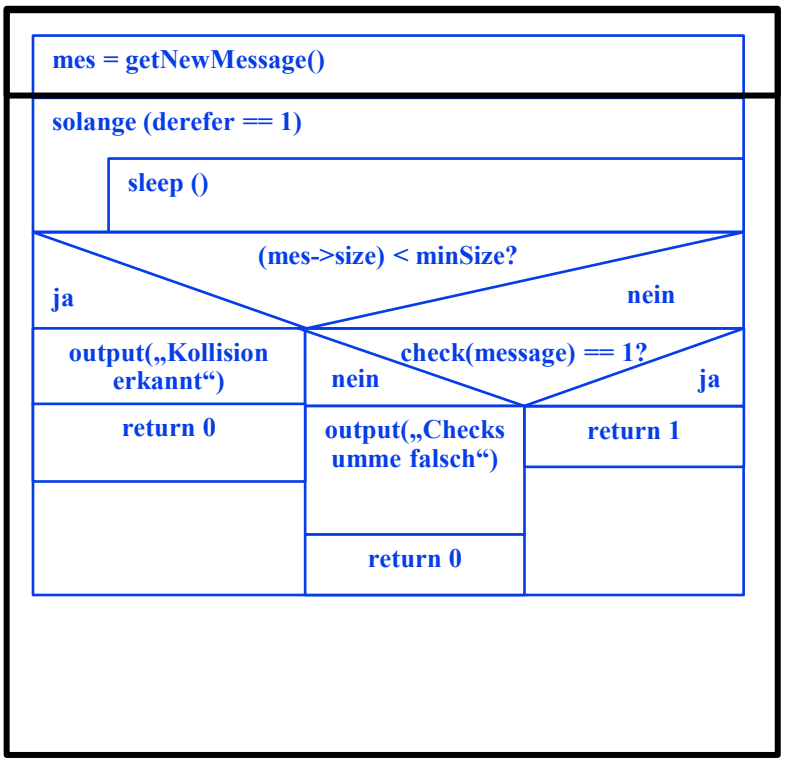
Punkte

f) Erstellen Sie ein **Nassi-Schneidermann-Diagramm** das folgenden Prozess beschreibt. Verwenden Sie C-Syntax und die C-Funktionen und Variablen die über dem Lösungskasten beschrieben sind:

1. Der Prozess weist der Variable `message *mes` die die aktuelle Nachricht zu.
2. Solange der Aufschub von Frames aktiviert ist, soll der Prozess warten.
3. Falls die Größe der Nachricht (`mes->size`) kleiner als die Minimalgröße ist wird „Kollision erkannt“ ausgegeben und 0 zurückgegeben ansonsten weitergegangen.
4. Falls die Checksumme korrekt ist, soll der Prozess 1 zurückgeben, ansonsten „Checksumme falsch“ ausgeben und 0 zurückgeben.

```
void output(char* out) // schreibt die Ausgabe out
message* getNewMessage() // gibt die aktuelle Nachricht zurück
int derefer // 1 falls der Aufschub aktiviert ist,0 sonst
void sleep() // wartet 10 Millisekunden
int minSize // Minimalgröße einer Nachricht
int check(message* mes) // 1 falls Checksumme von mes OK, 0 sonst
```

Nassi-Schneidermann-Diagramm



Vorname, Name _____ Matrikelnummer _____

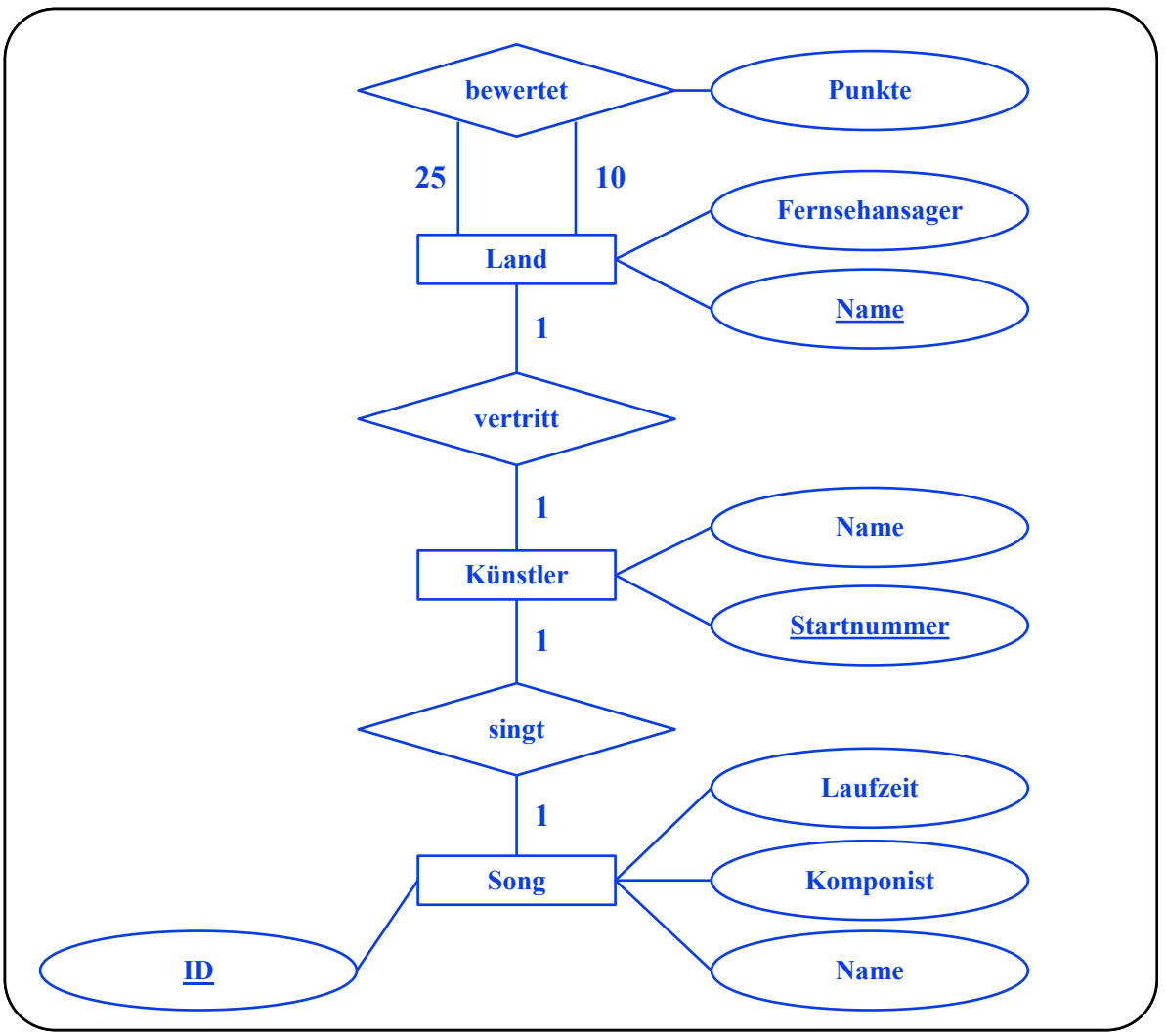
Punkte

Aufgabe DB: Datenbanken

Aufgabe DB: 30 Punkte

a) Modellierung eines Entity-Relationship-Diagramms:
Für den Eurovision Song Contest 2011 möchte die ARD zur Auswertung des Televotings eine Datenbank anlegen. Diese soll einen Überblick über die teilnehmenden Länder und deren Punktevergaben untereinander sowie über die Künstler mit ihren Songs geben.
Folgende Randbedingungen gelten: Ein *Künstler* tritt mit einem einzigen *Song* beim Contest an. Der *Künstler* vertritt hierbei genau ein *Land*. Im Anschluss bewerten die 25 teilnehmenden *Länder* jeweils ihre 10 favorisierten *Länder* mit einer unterschiedliche Anzahl von *Punkten.*
Folgende Attribute sind gegeben:
- Land: *Name, Fernsehansager*
- Künstler: *Name, Startnummer*
- Song: *Name, Komponist, Laufzeit*

Zeichnen Sie ein vollständiges Entity-Relationship-Diagramm. Wählen Sie aus den vorgegebenen Attributen sinnvolle Primärschlüssel und ergänzen Sie, falls zwingend erforderlich, fehlende Attribute. Fügen Sie die entsprechenden Kardinalitäten an.



Vorname, Name _____ Matrikelnummer _____

Punkte

Hinweis:
Für alle weiteren Teilaufgaben ist folgender unvollständiger Datenbankausschnitt gegeben.

Land		
Name	**Fernsehansager**	**Künstler**
Deutschland	Hape Kerkeling	6
Norwegen	Anne Rimmen	7
Irland	Derek Mooney	4
Finnland	Johanna Pirttilahti	5

Künstler		
Startnummer	**Name**	**Song**
5	Lordi	1
6	Lena	2
7	Alexander Rybak	3

bewertet		
VergabeLand	**EmpfängerLand**	**Punkte**
Norwegen	Deutschland	12
Finnland	Deutschland	10
Deutschland	Finnland	6
Norwegen	Irland	7
Deutschland	Irland	3

Song			
ID	**Name**	**Komponist**	**Laufzeit**

b) Legen Sie mit einem SQL-Befehl die Tabelle *Song* an. Befüllen Sie diese Tabelle anschließend mit einem SQL-Befehl mit folgenden Song, der von *Lena* gesungen wird:
- Name: *Satellite*
- Komponist: *Julie Frost*
- Laufzeit: *2,92*

```
CREATE TABLE Song (ID INTEGER PRIMARY KEY,
                   Name VARCHAR(25),
                   Komponist VARCHAR(25),
                   Laufzeit FLOAT);
```

```
INSERT INTO Song (ID, Name, Komponist, Laufzeit)
VALUES (2, "Satellite", "Julie Frost", 2.92);
```

Vorname, Name _____ Matrikelnummer _____

Punkte

c) Geben Sie den SQL-Befehl an, mit dem Sie sich die *Punkte* anzeigen lassen können, die die Fernsehansagerin *Anne Rimmen* bei der Live-Schaltung für *Lena* verkündet hat. Welche Antwort erhalten Sie bei den gegebenen Tabellen?

```
SELECT Punkte FROM bewertet WHERE VergabeLand =
   (SELECT Name FROM Land WHERE Fernsehansager = ”Anne Rimmen”)
AND EmpfängerLand =
   (SELECT Name FROM Land WHERE Künstler =
      (SELECT Startnummer FROM Künstler WHERE Name = ”Lena”)
   );
```

Anne Rimmen: “ **12** points for Lena!”