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

Vorname:	
Nachname:	
Matrikelnummer:	

Prüfung – Informationstechnik

Wintersemester 2012/13

08.03.2013

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

Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe GL: Zahlensysteme und logische Schaltungen

Aufgabe GL: 48 Punkte

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

1 (13)$_4$ = (_____)$_{10}$ = (_____)$_2$

2 (10,875)$_{10}$ = (_____)$_2$

b) Gegeben ist folgende IEEE 754 Gleitkommazahl:

1	1	0	1	1	0	0	1	1	0	0	1	1
V	e (4 Bit)				M (8 Bit)							

Tragen Sie Ihre Ergebnisse in die dafür vorgesehenen Textblöcke ein.
Wichtig: Ergebnisse und Nebenrechnungen außerhalb der dafür vorgesehenen Textblöcke werden nicht bewertet.

Vorzeichen: V

V =

Mantisse mit Vorkommastelle

M =

Bias: B

B =

Exponent: E

E =

Binäre Gleitkommazahl

$(Z)_2$ =

Dezimale Gleitkommazahl

$(Z)_{10}$ =

Vorname, Name _____ Matrikelnummer _____

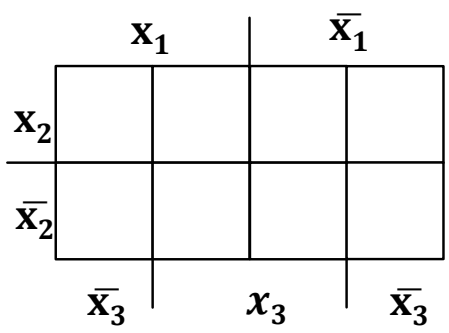
Punkte

Gegeben ist die folgende Wahrheitstabelle mit drei Eingängen und einem Ausgang.

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

c) Erstellen Sie die DNF *(Disjunktive Normalform)* aus der Wahrheitstabelle.

d) Minimieren Sie die DNF mit Hilfe des *KV-Diagramms* und schreiben Sie die minimierte Funktion in *boolescher Algebra* auf.

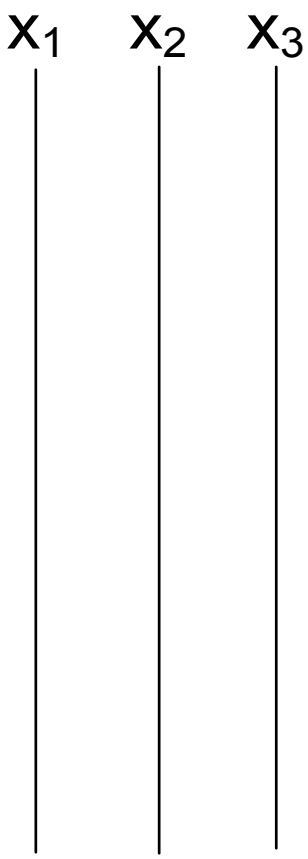


Vorname, Name _____ Matrikelnummer _____

Punkte

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

e) Zeichnen Sie die das *Blockschaltbild* der *KNF*.
Hinweis: Zur Ihrer Hilfe wurde die selbe Wahrheitstabelle oben noch mal abgedruckt.

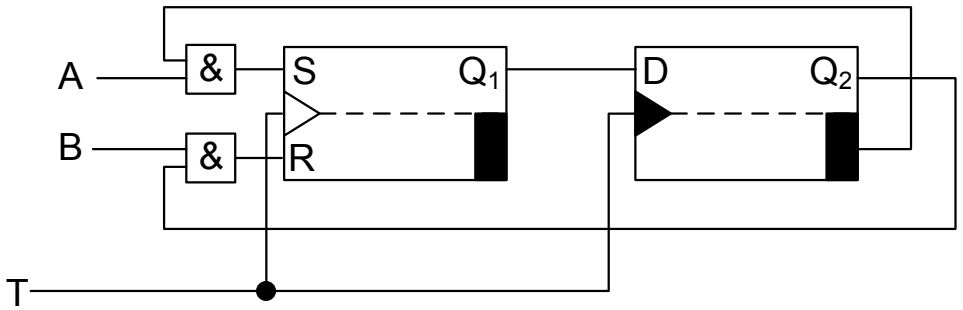


Vorname, Name

Matrikelnummer

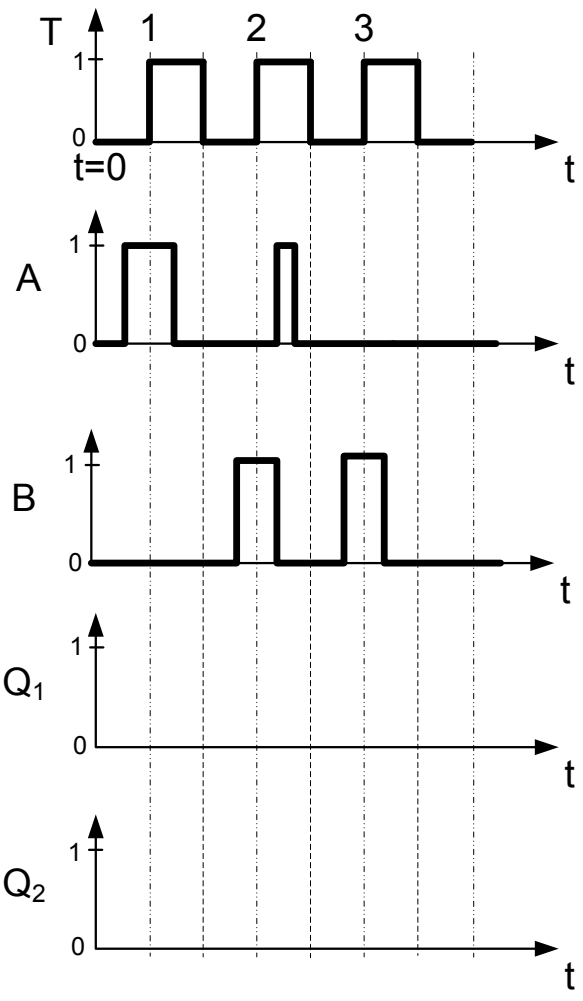
Punkte

Gegeben sei die folgende Master-Slave FlipFlop-Schaltung



Bei t = 0 seien alle Flip-Flops im folgenden Zustand: $Q_1 = 0$ und $Q_2 = 0$

f) Analysieren Sie die Schaltung, indem Sie die zeitlichen Verläufe für Q_1 und Q_2 in die vorgegebenen Koordinatensysteme eintragen. Die Signallaufzeiten können dabei vernachlässigt werden.



Vorname, Name

Matrikelnummer

Punkte

Aufgabe BS: Betriebssysteme

Aufgabe BS: 48 Punkte

1. Scheduling

Vier Prozesse (P1 bis P4) sollen mit einem Einkernprozessor abgearbeitet werden. Die Abbildung „Zeitverlauf der Prozesse“ gibt die Ausführungszeiten, d.h. die Zeiten, zu denen die Prozesse am Einkernprozessor eintreffen, und die Deadlines der einzelnen Prozesse an. Die vier Prozesse sollen zur Laufzeit mit unterschiedlichen Scheduling-Verfahren geplant werden. Für die Scheduling-Verfahren, bei denen feste Prioritäten berücksichtigt werden müssen, ist in der Tabelle „Prioritätenverteilung“ die entsprechende Prioritätenverteilung gegeben.

Prozess	Priorität
P1	1
P2	2
P3	3
P4	4

Tab.: Prioritätenverteilung (Prioritäts-level: „1“ = hoch, „4“ = niedrig)

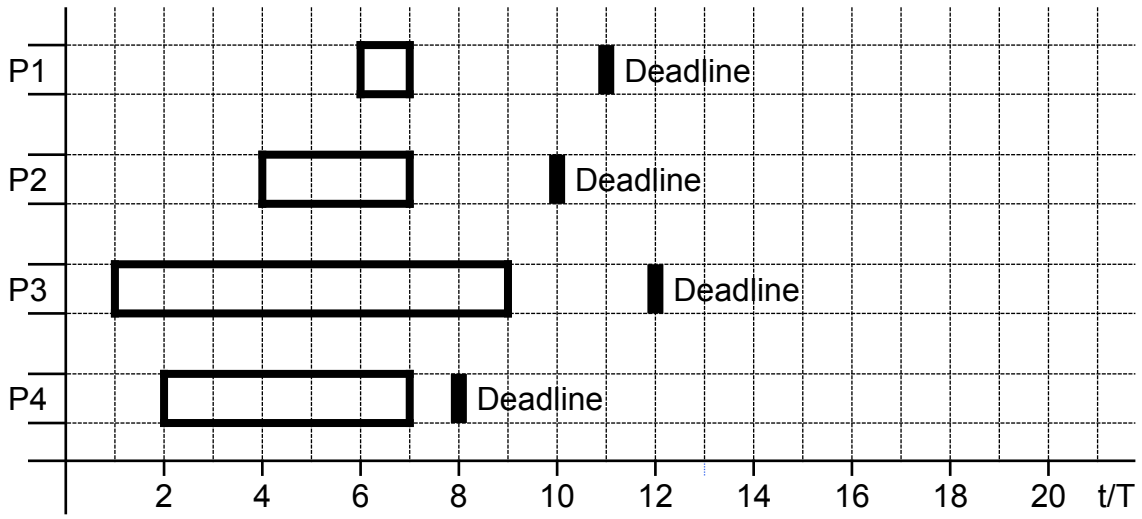


Abb.: Zeitverlauf der Prozesse

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

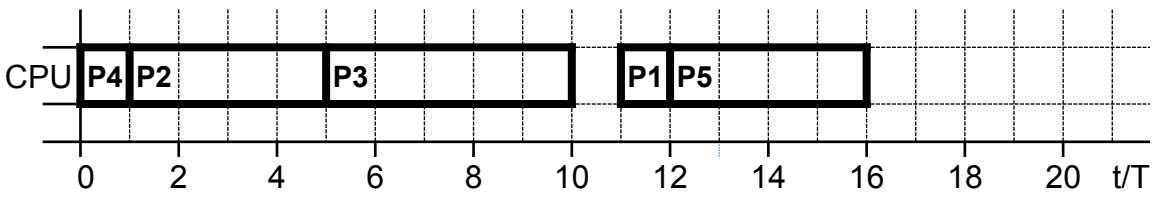


Diagramm: Lösungsbeispiel

Vorname, Name _____ Matrikelnummer _____

Punkte

a) In der ersten Teilaufgabe sollen die vier Prozesse präemptiv nach ihren Prioritäten abgearbeitet werden.

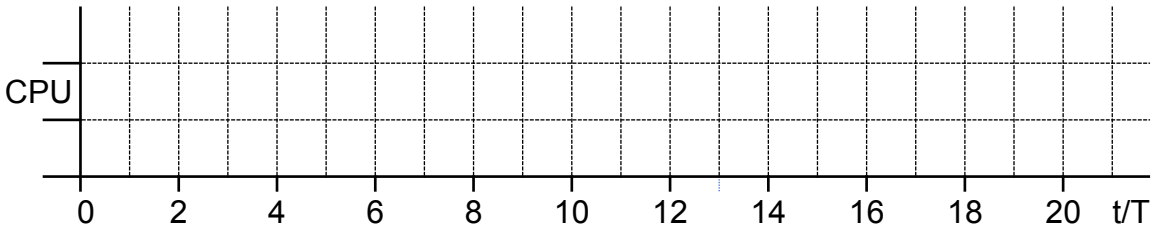


Diagramm: feste Prioritäten

b) In der zweiten Teilaufgabe erfolgt die Abarbeitung der oben gegebenen vier Prozesse präemptiv mit Hilfe des Round Robin-Verfahrens (RR) und 2T Zeitschlitzen.

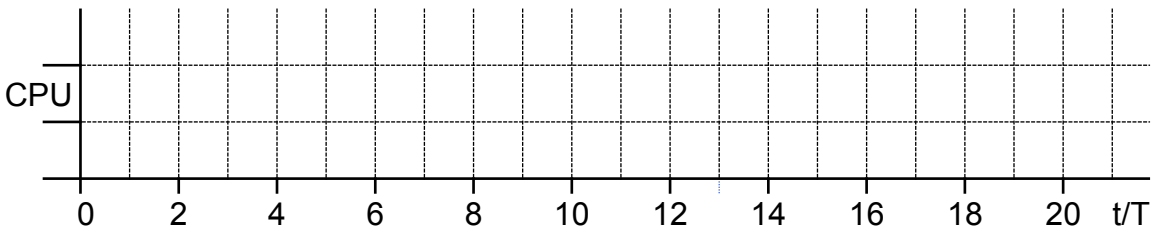


Diagramm: Round Robin (RR)

c) Markieren Sie in der Tabelle mit den Ziffern „1“ bzw. „5“, welches der fünf Scheduling-Verfahren am leichtesten („1“) bzw. am schwersten („5“) zu implementieren ist und welches der Verfahren sich am besten („1“) bzw. am schlechtesten („5“) für ein hartes Echtzeit-Scheduling eignet.

Verfahren / Merkmale	FIFO (First In First Out)	feste Prioritäten	LL (Least Laxity)	EDF (Earliest Deadline First)	RR (Round Robin)
Implementierungsaufwand					
Eignung für harte Echtzeit					

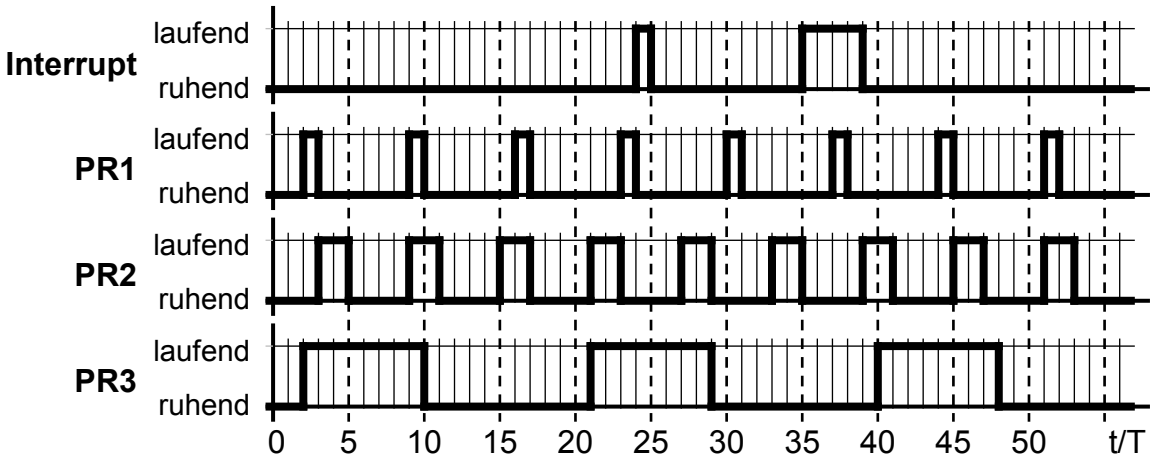
Vorname, Name _____ Matrikelnummer _____

Punkte

2. Asynchrone Programmierung

Drei periodische und präemptive Prozesse (PR1 bis PR3) sollen mit dem Verfahren der asynchronen Programmierung auf einem Einkernprozessor ausgeführt werden. Der Prozess PR1 besitzt die höchste, der Prozess PR3 die niedrigste Priorität. Die Ausführung wird durch zwei Interrupts unterbrochen. Tragen Sie in das unten angegebene Diagramm die tatsächliche Abarbeitung der Rechenprozesse nach dem Verfahren der asynchronen Programmierung ein.

Hinweis: Bei größeren Korrekturen verwenden Sie bitte das **Ersatzfeld** und markieren das zu wertende Lösungsfeld.



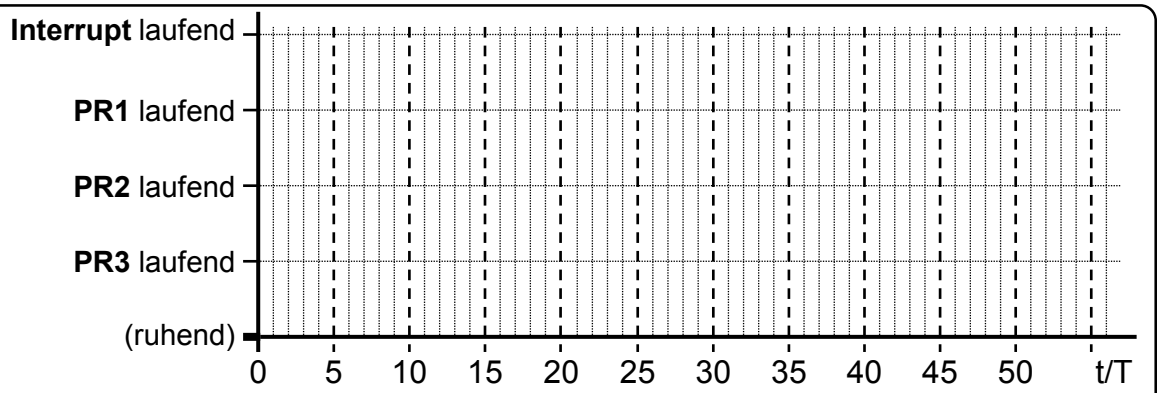


Diagramm: asynchrone Programmierung

☐ *Dieses Lösungsfeld werten! (hier ankreuzen)*

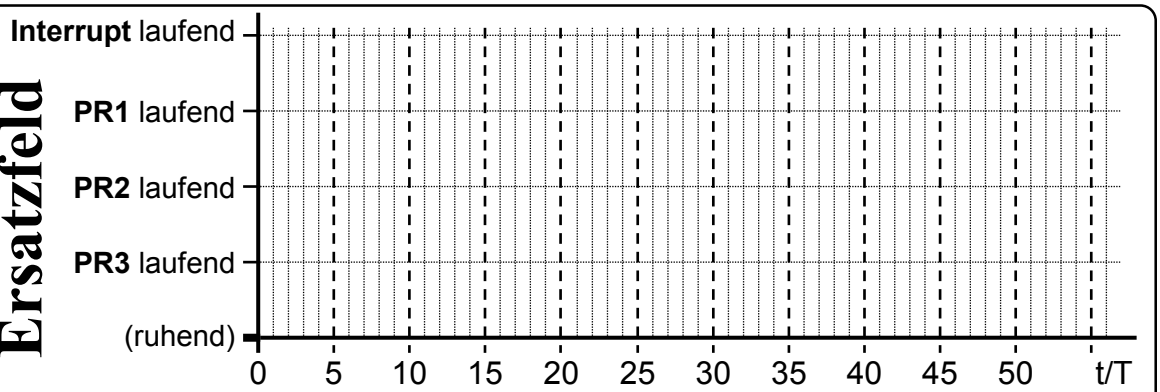


Diagramm: asynchrone Programmierung – **Ersatzfeld, falls oberes Feld falsch**

☐ *Dieses Lösungsfeld werten! (hier ankreuzen)*

Vorname, Name _____ Matrikelnummer _____

Punkte

3. Semaphoren

Gegeben ist die Anordnung von Semaphor-Operationen am Anfang und am Ende der Tasks A,B,C. Ermitteln Sie für die Fälle I und II, ob und in welcher Reihenfolge diese Tasks bei der angegebenen Initialisierung der Semaphor-Variablen ablaufen. Geben Sie zusätzlich an, ob sich die Taskreihenfolge wiederholt bzw. um welche Art von Verklemmung es sich handelt.

Tasks:	**A**	**B**	**C**
Taskabarbeitung ↓	P(S1) P(S2) P(S3) P(S3) …	P(S1) P(S2) P(S2) … V(S1) V(S1) V(S3) V(S3)	P(S3) … V(S2) V(S3)

Hinweis:
Die für die Ausführung eines Tasks notwendigen Semaphoren sollen nur im Block verwendet werden (z.B. Task A startet nur, wenn alle Semaphoren S1, S2, S3, S3 frei sind).
Sind mehrere Tasks ablauffähig, gelten folgende Prioritäten: A: hoch, B: mittel, C: niedrig. P(Si) senkt Si um 1, V(Si) erhöht Si um 1. Geben Sie die Reihenfolge der ablaufenden Tasks in folgender Schreibweise an: _____
z.B. „ABCABB → Wiederholung“.

Fall I:

	Semaphoren		
Task	**S1**	**S2**	**S3**
Start	3	2	5

Fall I: _____

Fall II:

	Semaphoren		
Task	**S1**	**S2**	**S3**
Start	2	0	5

Fall II: _____

Vorname, Name

Matrikelnummer

Punkte

Aufgabe MSE: Modellierung und Systementwicklung

Aufgabe MSE: 48 Punkte

1. Strukturierte Analyse / Real Time

Modellieren Sie einen Bankautomat mit Hilfe der Strukturierten Analyse/ Real Time.

a) Erstellen Sie zunächst das **Datenkontextdiagramm** für den Prozess *Geld abheben*. Dabei sollen sowohl der Kunde, als auch die Bank als externe Systeme betrachtet werden. Der Kunde kann die Benutzereingaben *EC-Karte* und *PIN-Code* an den Geldautomaten schicken und bekommt von diesem *Geld* und seine *EC-Karte* zurück. Die Bank empfängt *Abbuchungsaufträge* und gibt *Freigaben* zurück. Achten Sie dabei auf eine korrekte Beschriftung.

Vorname, Name _____ Matrikelnummer _____

Punkte

b) Im weiteren Verlauf der Modellierung wurde der Prozess *Geld abheben* aus Aufgabe a) mit Hilfe weiterer Datenflussdiagramme weiter verfeinert. Dabei wurde ein nicht mehr weiter zu zerlegender Prozess *PIN überprüfen* erstellt. Für diesen soll nun eine **Prozessspezifikation (PSPEC)** erstellt werden.
Vervollständigen Sie die unten abgebildete PSPEC des Prozesses *PIN überprüfen* des Geldautomaten. Um Geld abheben zu können, muss die Karte eingelesen und der PIN überprüft werden. Hierfür sind die Eingaben *PIN_eingegeben* sowie *Karte_eingelesen* erforderlich. Wenn die Karte eingelesen und die PIN eingegeben ist, wird die *PIN mit den Kartendaten verglichen*. Abhängig von der Übereinstimmung wird ausgegeben, ob die eingegebene PIN richtig war (*PIN_richtig*) oder nicht (*PIN_falsch*).
Tragen Sie in die PSPEC alle Ein- und Ausgänge ein. Geben Sie außerdem eine kurze Beschreibung in Pseudo-Code, wie die Ausgänge anhand der Eingänge gesetzt werden.

PSPEC 4: PIN überprüfen

INPUTS:

OUTPUTS:

TRANSFORMATION (in Pseudo-Code):

Vorname, Name _____ Matrikelnummer _____

Punkte

c) Da die Funktionalität des Bankautomaten aus Aufgabe a) für den Alltagseinsatz nicht ausreicht, soll diese nun erweitert werden. Der Kunde soll nun zwischen den folgenden Funktionen wählen können:

- Geld abheben
- Finanzstatus zeigen
- Vorgang Abbrechen

Zur Auswahl der Funktionalität stehen dem Kunden insgesamt drei Tasten zur Verfügung: Taste *Geld*, Taste *Status*, Taste *Abbrechen.*

Es gelten die folgenden Regeln:

- Sobald die Taste *Abbrechen* gedrückt wird, wird der Vorgang abgebrochen.
- Wird eine der Tasten *Geld* oder *Status* gedrückt, so wird die entsprechende Funktion ausgeführt, aber nur wenn keine weitere Taste gedrückt ist.

Erstellen Sie die **Kontrollflussspezifikation (CSPEC)** in Form einer Entscheidungstabelle anhand welcher entschieden wird, welche Funktionalität ausgeführt wird.

	R1	R2	R3
Taste Geld			
Taste Status			
Taste Abbrechen			
Vorgang abbrechen			
Geld abheben			
Finanzstatus zeigen			

Vorname, Name

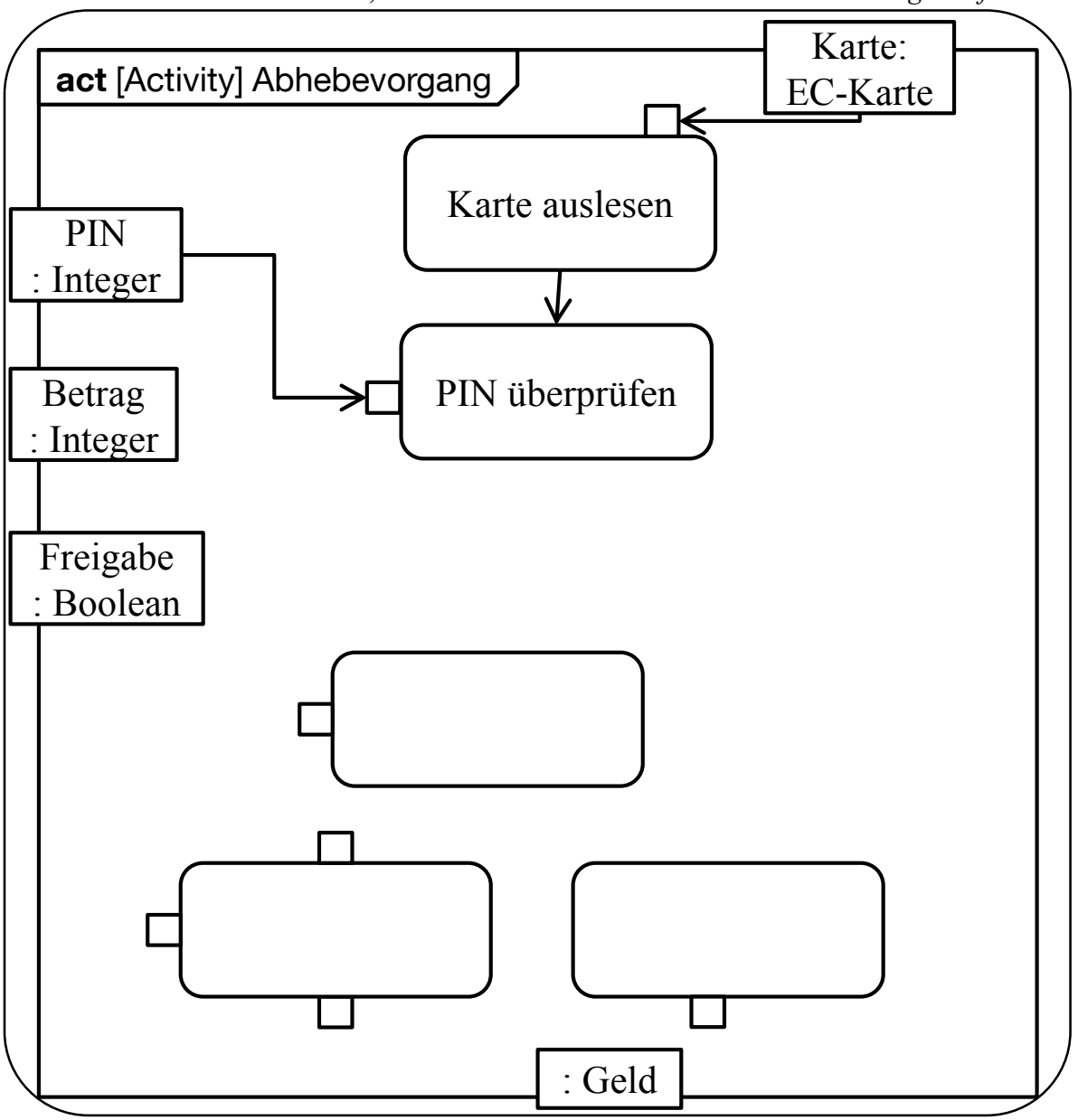
Matrikelnummer

Punkte

2. SysML

a) Für einen Geldautomaten soll nun ein SysML **Aktivitäts-Diagramm** erstellt werden, welches den Abhebevorgang beschreibt. Verzichten Sie hierbei auf die Angabe von Datentypen an Pins und erstellen Sie keine zusätzlichen Pins oder Aktivitäten. Zeichnen Sie an geeigneten Stellen Start- und Endknoten ein.

- Der Automat wartet darauf, dass eine EC-Karte eingesteckt wird. Sobald dies der Fall ist, wird die *Karte ausgelesen*.
- Anschließend wird in der Aktivität *PIN überprüfen* eine eingegebene PIN mit den ausgelesenen Kartendaten verglichen.
- Wurde die PIN falsch eingegeben, so wird die *Karte ausgeworfen*.
- Wurde die PIN korrekt eingegeben, so wird der *abzuhebende Betrag* abgefragt, welcher anschließend weitergegeben wird.
- Kommt die *Freigabe* von der Bank, so wird dem Kunden sein *Geld ausbezahlt*.
- Ist das Geld ausbezahlt, so wird die EC-Karte des Kunden wieder *ausgeworfen*.

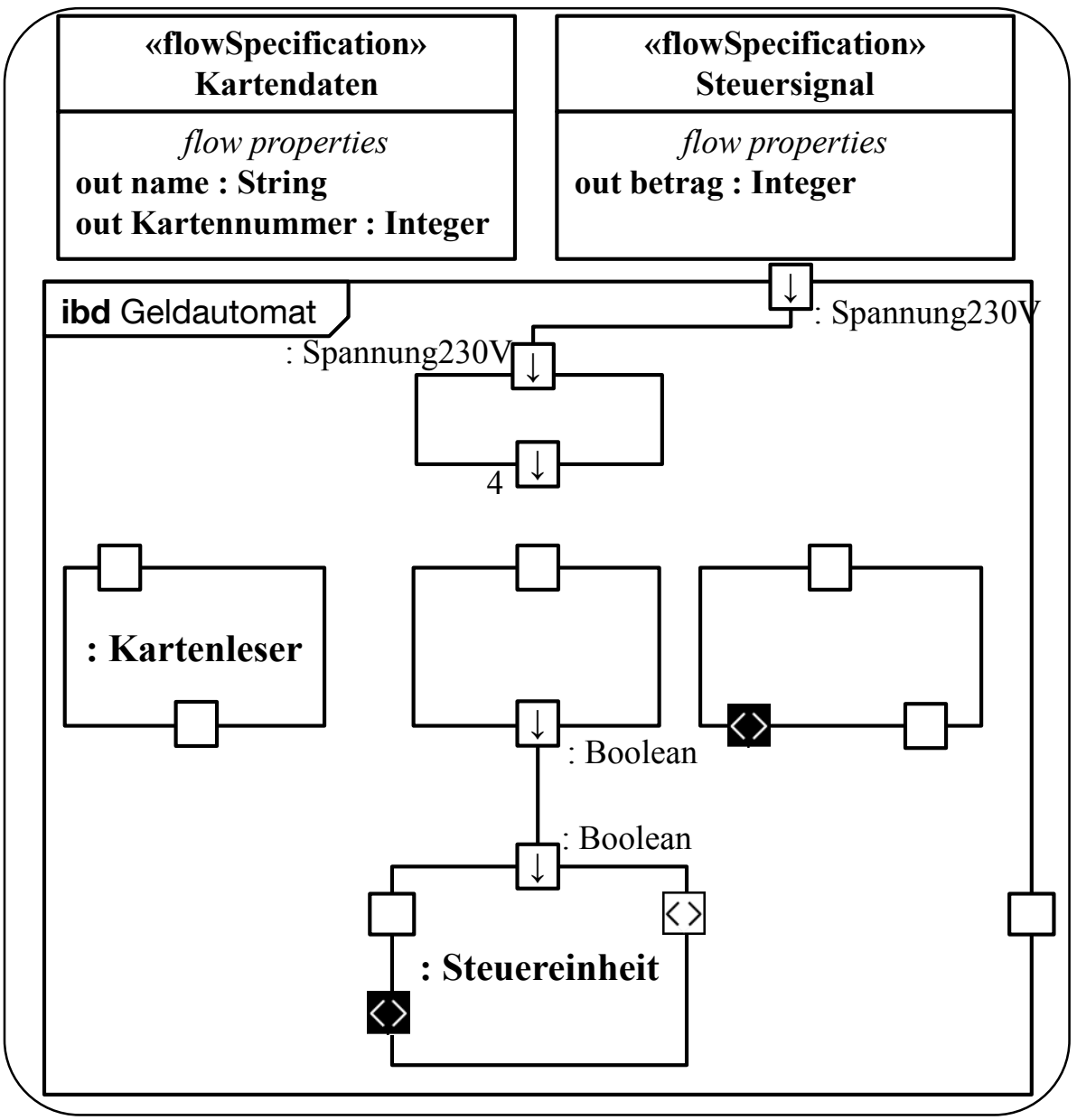


Vorname, Name _____ Matrikelnummer _____

Punkte

b) Erstellen Sie ein SysML **Internes Blockdiagramm** und vervollständigen Sie die FlowSpecifications für den Geldautomaten. Verzichten Sie hierbei auf eine Namensvergabe für Komponenten und Flüssen.

- Der *Geldautomat* wird von außen mit *Spannung230V* versorgt. Dieser wird intern von einem *Netzteil* in *Spannung12V* umgewandelt und an alle anderen Komponenten verteilt. Behandeln Sie die Stromflüsse jeweils als elementare Flüsse.
- Das *Tastenfeld* sendet einen Booleschen Wert an die *Steuereinheit*.
- Der *Kartenleser* sendet eine komplexe Datenstruktur an die *Steuereinheit*, welche mit der FlowSpecification *Kartendaten* beschrieben wird.
- Die *Geldausgabe* und die *Steuereinheit* kommunizieren mit Hilfe eines *Steuersignals,* wobei die *Steuereinheit* den Betrag übermittelt und von der *Geldausgabe* einen Status zurückbekommt. Des Weiteren gibt die Geldausgabe *Geld* (elementarer Datentyp) nach außen ab.



Vorname, Name _____ Matrikelnummer _____

Punkte

Aufgabe C: Programmieren in C

Aufgabe C: 96 Punkte

Teil 1: Datentypen, Ein-/Ausgabe und Bool'sche Algebra

Sie bekommen von einem Parkhausbetreiber den Auftrag, ein Programm zu schreiben, mit dem die Parkgebühren berechnet werden können.

a) Legen Sie zunächst je eine Variable an für

- die Zahl des Wochentags (beginnend mit Montag als erstem Tag der Woche)
- den Grund-Preis pro Stunde in Euro mit Cent-Beträgen
- die Gesamtzahl parkender Fahrzeuge im **laufenden Jahr** (>1000 Fahrzeuge/Tag!)

und **initialisieren** Sie die Variablen mit sinnvollen Werten.

b) Der Betreiber soll die Möglichkeit haben, einen Preisnachlass auf den Grundpreis pro Stunde flexibel einzustellen. Dabei wird der neu eingeführten Variable ***fNachlass*** eine Fließkommazahl zugewiesen. Danach soll der resultierende Preis zur Überprüfung auf **drei Nachkommastellen und drei Vorkommastellen mit führenden Nullen** (z.B. „001.120“) genau ausgegeben werden. Füllen Sie die Lücken im Quellcode.

```
_________ fNachlass  = 0; //Anlegen der Variable für den Nachlass
float fPreis = 1.80;

//Eingabe
printf ("Bitte geben Sie den Nachlass in Euro ein: ");

__________ (_____________________);

fPreisNachlass = fPreis - fNachlass;

//Ausgabe
printf ("\n\nIhr Preis lautet: ________\n", fPreisNachlass);
```

c) Werten Sie folgende Ausdrücke aus und schreiben Sie das Ergebnis in die rechte Spalte.
Int-Variablen: A = 2; D = 3; Float-Variablen: B = 0.6; C = π;

(D && (int)B) != !A	
((A && D) | (int)C)	
(A & !D)	

Vorname, Name _____ Matrikelnummer _____

Punkte

d) Das unten aufgeführte Programm berechnet die Parkgebühr in Abhängigkeit des Grundpreises, der Parkdauer sowie des Nachlasses für verschiedene Tage. Markieren Sie alle **6 Syntaxfehler** und die entsprechende Zeile.

Beispiel: `int iHilf = 0` ☒

```
#include <stdio.h>
int main {
    float fPreis     = 1.50;     //Grundpreis pro Stunde
    float fNachlass  = 0.2;      //Nachlass auf Grundpreis
    iParkdauer       = 0;        //Anzahl angebrochener Stunden
    float fFeiertags = 1;        //Grundpreis Feiertags
    int   iHilf      = 0;

    printf("Parkdauer in Stunden: ");
    if(scanf("i",&iParkdauer)==0)

        printf("Unzulaessige Eingabe!");
        return 1;
    }

    fPreis = iParkdauer * fPreis;
    printf("\nPreis an Werktagen: %f EURO",fPreis);

    fPreis -= (--iParkdauer)*fNachlass*fPreis
    printf("\nPreis am Wochenende: %f EURO",fPreis);

    iHilf = --iParkdauer/2;
    fPreis = iHilf * fFeiertags * iParkdauer;
    printf("\nPreis an Feiertagen: %i EURO",(int)fPreis);

return true;
}
```

e) Geben Sie die Ausgabe des Programms an, wenn bei der Eingabe der **Wert „3“** eingegeben wird. Gehen Sie jetzt davon aus, dass der Code von Fehlern bereinigt wurde und die Ausgaben der beiden float-Variablen auf **zwei Nachkommastellen** gerundet wurden.

```
Preis an Werktagen:    _____ Euro

Preis am Wochenende:   _____ Euro

Preis an Feiertagen:   _____ Euro
```

Vorname, Name _____ Matrikelnummer _____

Punkte

Teil 2: Kontrollstrukturen

a) Das Programm soll so erweitert werden, dass der Betreiber den Parkpreis pro Stunde anhand der Länge *fLaenge* und der Anzahl der Sitzplätze (laut Fahrzeugschein) *iSitze* des Fahrzeugs bestimmen kann. Der Preis pro Stunde soll folgendermaßen auf Basis des Grundpreises ermittelt werden:

- Für Fahrzeuge die länger als 5 Meter sind oder mehr als 7 Sitzplätze haben, soll der Grundpreis um 20% erhöht werden.
- Bei Fahrzeugen mit nur 4 Sitzplätzen oder weniger, welche kürzer als 4,5 Meter sind, soll der Grundpreis um 20% gesenkt werden.
- für alle anderen Fahrzeuge wird der Grundpreis um 10% gesenkt.

Gehen Sie davon aus, dass der Code zum Einlesen von Länge und Anzahl der Sitze bereits realisiert wurde. Verändern Sie die Variable für den Parkpreis pro Stunde *fPreis* für die genannten Fälle. Nutzen Sie hierfür eine **IF-ELSE** Abfrage.

```
#include <stdio.h>
int main(){
    float fPreis  = 1.50;     //Grundpreis pro Stunde
    float fLaenge;            //Laenge des Autos
    int   iSitze;             //Anzahl Sitze

    //Eingabe der Laenge mit externer Funktion
    fLaenge=fEinlesen();
    //Eingabe der Anzahl der Sitze mit externer Funktion
    iSitze=iEinlesen();

    //Preisberechnung mit IF-ELSE




    printf(„Parkpreis: %.2f € \n", fPreis );
    return 0;
}
```

Vorname, Name _____ Matrikelnummer _____

Punkte

b) Der Betreiber des Parkhauses möchte, dass das Programm abhängig vom Wochentag unterschiedliche Öffnungszeiten ausgibt. Das Einlesen der Benutzereingabe in die Variable *wtag* ist bereits von einer externen Funktion realisiert.

- Definieren Sie zunächst einen neuen Datentyp WTAG mit Hilfe von **ENUM**, welcher aus Abkürzungen der 7 Wochentage bestehen soll (z.B. „MO“ für Montag).
- Durch eine Struktur sollen nun die verschiedenen Öffnungszeiten ausgegeben werden. Für die Tage Montag, Dienstag und Donnerstag soll „6-20 Uhr“, für Mittwoch „6-24 Uhr“ und an übrigen Tagen „Geschlossen!“ ausgegeben werden.

Implementieren Sie die **SWITCH-CASE-Abfrage** so, dass die drei benötigten Print-Anweisungen jeweils nur einmal verwendet werden müssen.

```
#include <stdio.h>
//Anlegen des Variablentyp WTAG als enum
```

```
int main(){
    WTAG wtag;  //Variable vom Typ WTAG anlegen

    //Eingabe durch den Betreiber mit externer Funktion
    wtag = wtagEinlesen();

    //Ausgabe der Öffnungszeiten abhaenging vom Wochentag
```

```
    return 0;
}
```

Vorname, Name _____ Matrikelnummer _____

Punkte

Teil 3: Erweiterte Datentypen

a) Der Parkhausbetreiber möchte nun die Auslastung beobachten. Für jedes der drei Stockwerke soll eine **ganzzahlige Prozentzahl** Auskunft über das Verhältnis von genutzten zu freien Parkplätzen geben.

Deklarieren Sie ein Array von geeignetem Typ und geeigneter Länge und benennen Sie es sinnvoll. **Initialisieren** Sie dann die erste und die letzte Ebenen mit 10% bzw. 20%.

b) Um ein noch genaueres Bild über die Nutzung des Parkhauses zu bekommen, soll nun jeder einzelne Parkplatz erfasst werden. Jede der drei Parkebenen besitzt 50 Stellplätze und jeder einzelne Stellplatz kann belegt „B“ oder frei „F“ sein. **Deklarieren** Sie ein Array von geeignetem Datentyp, Länge und Dimension, sodass jeder Parkplatz über Stockwerk und Stellplatznummer (im jeweiligen Stockwerk) eindeutig ist. **Initialisieren** Sie den ersten Stellplatz im ersten Stockwerk als **frei** und den letzten Stellplatz im obersten Stock als **belegt**.

Vorname, Name _____ Matrikelnummer _____

Punkte

Teil 4: Schleifen

Zur Vervollständigung vorgegeben ist ein Programm zur Bestimmung der aktuellen Belegung bzw. der Anzahl von freien Stellplätzen in einem Parkhaus. Das Parkhaus besitzt **Stockwerke (Decks)** und in jedem dieser Stockwerke **Stellplätze (Slots)**. Die Belegung wird von einer externen Funktion, welche auf die Sensorik im Parkhaus zugreift, in das Array *iParkhaus* eingelesen. Der Inhalt des Array sieht z.B. wie folgt aus:

Decks /Slots	**0**	**1**	**2**	**3**	**4**	**5**	**…**	**59**
4	1	0	0	0	1	0	…	0
3	0	0	1	1	1	1	…	1
⋮	…	…	…	…	..	..	⋱	…

0 entspricht frei
1 entspricht belegt

Vervollständigen Sie die Initialisierung von *iParkhaus* mit den bereits vorgegebenen Präprozessor-Variablen. Verwenden Sie für das Zählen aller belegten Stellplätze eine **FOR-Schleife** und eine **WHILE-Schleife**, vervollständigen Sie weiterhin die zugehörige **IF-Bedingung**, welche den Wert von *iBelegt* pro Zählvorgang um den Wert 1 erhöht. Vervollständigen Sie außerdem die fehlende Berechnung des Ergebniswertes.

```
#include <stdio.h>
#define DECKS 5
#define SLOTS 60
int main()
{
int i=0;
int j=0;
int iBelegt=0;
int iParkhaus _____ ={0};

// Aktuelle Belegung des Parkhaus einlesen
iParkhaus=SensorStellplatzEinlesen();

for(_____)    // Decks abtasten
    {

        WHILE(_____) // Slots abtasten
        {
        if(_____) iBelegt++;


        _____
        }
    }
    printf("\n");

printf("\nLeere Parkbuchten: %i", _____);
return 0;
}
```

Vorname, Name

Matrikelnummer

Punkte

Teil 5: Zeiger

Gegeben sei ein kurzes C-Programm, welches ein Array mit vier Elementen und die Zeigervariable *pizeiger* initialisiert. Die darauf folgenden Befehle führen jeweils zur einer Manipulation des Zeigers sowie der Werte im Array. Geben Sie die veränderten Werte des Arrays nach dem Ausführen des Codes an. Kreuzen Sie in der mittleren Spalte an, wo sich der Zeiger am Ende des Codes befindet.

```
#include <stdio.h>
int main ()
{
    int *pizeiger=NULL;
   int Ai[4]={2,9,4,7};
   pizeiger=Ai;
   *pizeiger=4;
   pizeiger=&Ai[1]+1;
   *(pizeiger)+=*(pizeiger-1);
   *(pizeiger-1)=Ai[0]%3;
   *(++pizeiger)=*pizeiger+2*2;
   return 0;
}
```

Array-Element	Zeiger	Wert des Elementes
Ai[0]		
Ai[1]		
Ai[2]		
Ai[3]		

Vorname, Name _____ Matrikelnummer _____

Punkte

Teil 6: Funktionen mit Call-by-reference und Sortierung

Die Buchstabenbezeichnungen eines VIP-Parkdecks mit 10 Plätzen ist durcheinander geraten. Gegeben sei daher ein C-Programm mit zwei Call-by-Reference Funktionen, welche ein **Array** *cDeck* mit der Länge *ANZAHL* und der zufälligen Buchstabenreihenfolge sortieren soll. Die Main-Funktion ist nicht dargestellt. Die Reihenfolge soll dabei **aufsteigend bzw. von „A bis Z“** sortiert werden. Die Funktion für das Vertauschen von zwei Plätzen im Array sowie die Funktion für die Sortierung sollen zunächst in ihrer Deklarierung und des Weiteren in ihrer Funktionalität vervollständigt werden. Beachten Sie die Kommentare im Code.

```
#include <stdio.h>
#define ANZAHL 10        //Anzahl der Parkslots

//1. Tauscht die Plätze zweier Parkslot-Buchstaben

__________ Vertausche (__________ a, _____________ b)
{
    char temp;

    _____________________//Zwischenspeichern

    _____________________

    _____________________

}
//2. Sortiert nach Buchstaben, beginnend mit A

___________ ParkSort (_____________ cDeck)
{
    int i=0;
    int iHilf=1;

    while (___________________) // Schleife ausführen
          //solange Tauschvorgänge stattgefunden haben
    {
        iHilf=0; //Hilfsvariable zurücksetzen

        for (_____________________________________)
         // Schleife welche jedes Element des Array mit
         // seinem Nachbarn vergleicht
        {
            if (cDeck[i] _________ cDeck[i+1])
            {

              Vertausche(_____________________________);
              iHilf++;
            }
        }
    }} (...)
```