Musterlösung

Klausur Softwaretechnik

11. Oktober 2001

Name: _____

Vorname: _____

Matrikelnummer: _____

Aufg.0	Aufg.1	Aufg.2	Aufg.3	Aufg.4	Aufg.5	Σ	Note
1	14	11	6	12	16	60	1.0

Die Klausur besteht aus 11 Seiten und ist geheftet abzugeben.
Für die Vollständigkeit nicht gehefteter Klausuren wird keine Verantwortung übernommen.

Informationswirte müssen Aufgabe 5 und Aufgabenteil i) von Aufgabe 1 nicht bearbeiten!

Name: _____ Matrikelnummer: _____

Aufgabe 0 (1 Punkt)

Schreiben Sie auf jedes Blatt ihren Namen und Ihre Matrikelnummer in die dafür vorgesehenen Felder.

Aufgabe 1 (14 Punkte)

Lösen Sie folgende Aufgaben. (Bei den meisten Fragen sind mehrere Antworten anzukreuzen. Punkte gibt es nur für vollständig richtig beantwortete Aufgaben. Für falsche Antworten gibt es keinen Punktabzug.)

a) Das Entwurfsmuster Model/View/Controller (MVC) kombiniert welche Entwurfsmuster?

1P
nur, wenn komplett richtig!

- ☐ Besucher
- ☐ **Beobachter**
- ☐ **Kompositum**
- ☐ **Strategie**
- ☐ Vermittler
- ☐ Zuständigkeitskette

b) Was macht eine Bequemlichkeitsmethode so „bequem“?

1P
nur, wenn komplett richtig!

- ☐ Sie ist „statisch” deklariert und benötigt daher kein instanziiertes Objekt.
- ☐ Sie erlaubt die einheitliche Behandlung von Objekten, die unterschiedlichen Klassen angehören, aber gemeinsame Attribute oder Methoden besitzen.
- ☐ **Sie vereinfacht Methodenaufrufe durch die Bereitstellung häufig genutzter Parameterkombinationen.**
- ☐ Sie ermöglicht es Unterklassen, bestimmte Schritte eines Algorithmus zu überschreiben, ohne seine Struktur zu verändern.
- ☐ Keines der genannten.

c) Welche Punkte sind nach Balzert Bestandteile eines Lastenheftes?

1P
nur, wenn komplett richtig!

- ☐ **Zielbestimmung**
- ☐ **Produkteinsatz**
- ☐ Benutzer-Eigenschaften
- ☐ **Produktdaten**
- ☐ Entwicklungsumgebungen
- ☐ Abgrenzungskriterien

d) Welche Kategorien werden bei der Funktionspunktanalyse betrachtet?

1P
nur, wenn komplett richtig!

- ☐ **Eingabedaten**
- ☐ Querverweise
- ☐ **Ausgaben**
- ☐ **Referenzdaten**
- ☐ Anwendungsfälle

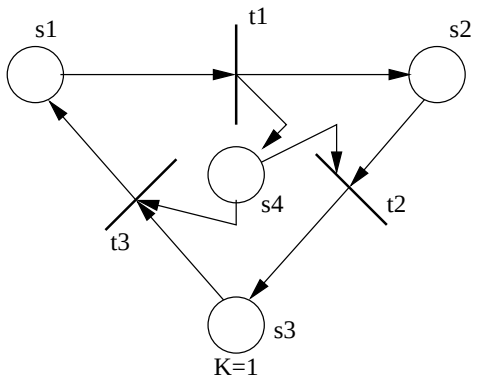
Name: _____ Matrikelnummer: _____

e) Ordnen Sie jedes der folgenden Dokumente der Entwicklungsphase zu, deren Ergebnis es im Idealfall ist.

2P
-0,5P pro falsche Zeile

Dokument	Phase			
	Planungs-	Definitions-	Entwurfs-	Implementierungs-
Projektkalkulation	x			
Spez. der Systemkomponenten			x	
Durchführbarkeitsstudie	x			
Softwarearchitektur			x	
Pflichtenheft		x		
Quellprogramme				x

f) Gegeben sei folgendes Petrinetz *P* mit Anfangsmarkierung *M* = (2,0,0,0).



Das Netz *P* ist:

- ☐ lebendig.
- ☒ **nicht lebendig.**
- ☐ tot.
- ☒ **beschränkt.**
- ☐ unbeschränkt.

1,5P
nur, wenn komplett richtig!

g) Im Petrinetz *P* auf Aufgabenteil g) sei nun W((t3,s1)) = 2 mit Anfangsmarkierung *M*. Das modifizierte Netz *P'* ist:

- ☒ **lebendig**.
- ☐ nicht lebendig.
- ☐ tot.
- ☐ beschränkt.
- ☒ **unbeschränkt**.

1,5P
nur, wenn komplett richtig!

Name: _____ Matrikelnummer: _____

h) Gegeben sei folgende Entscheidungstabelle:

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16
B1	J	J	J	J	J	J	J	J	N	N	N	N	N	N	N	N
B2	J	J	J	J	N	N	N	N	J	J	J	J	N	N	N	N
B3	J	J	N	N	J	J	N	N	J	J	N	N	J	J	N	N
B4	J	N	J	N	J	N	J	N	J	N	J	N	J	N	J	N
A1	X	X	-	X	X	X	-	X	X	X	-	X	X	X	-	X
A2	-	X	X	X	-	X	X	-	-	X	-	X	-	X	-	-
A3	-	-	X	-	-	X	X	X	-	-	X	-	-	X	X	X

Konsolidieren Sie die Entscheidungstabelle und tragen Sie das Ergebnis in folgende Tabelle ein:

	R1'	R2'	R3'	R4'	R5'	R6'
B1	-	-	-	-	J	N
B2	-	J	N	N	-	-
B3	J	-	N	J	N	N
B4	J	N	N	N	J	J
A1	X	X	X	X	-	-
A2	-	X	-	X	X	-
A3	-	-	X	X	X	X

3P

0,5P pro richtige Spalte

Der folgende Aufgabenteil ist von **Informationswirten** nicht zu bearbeiten!

i) Betrachten Sie folgende Ausschnitte aus einem C-Programm zur Matrixmultiplikation. Welcher Ausschnitt ist bezüglich der Cache-Benutzung der günstigste? Nehmen Sie eine zeilenweise Speicherabbildung (wie in C) an und gehen Sie weiterhin davon aus, dass das Programmstück nicht von einem Übersetzer weiter optimiert wird.

2P

□
```
for (i=0; i<SIZE; i++)
    for (j=0; j<SIZE; j++)
        for (k=0; k<SIZE; k++)
            c[i][j] += a[i][k] * b[k][j];
```

□
```
for (i=0; i<SIZE; i++)
    for (j=0; j<SIZE; j++)
        for (k=0; k<SIZE; k++)
            c[k][i] += a[k][j] * b[j][i];
```

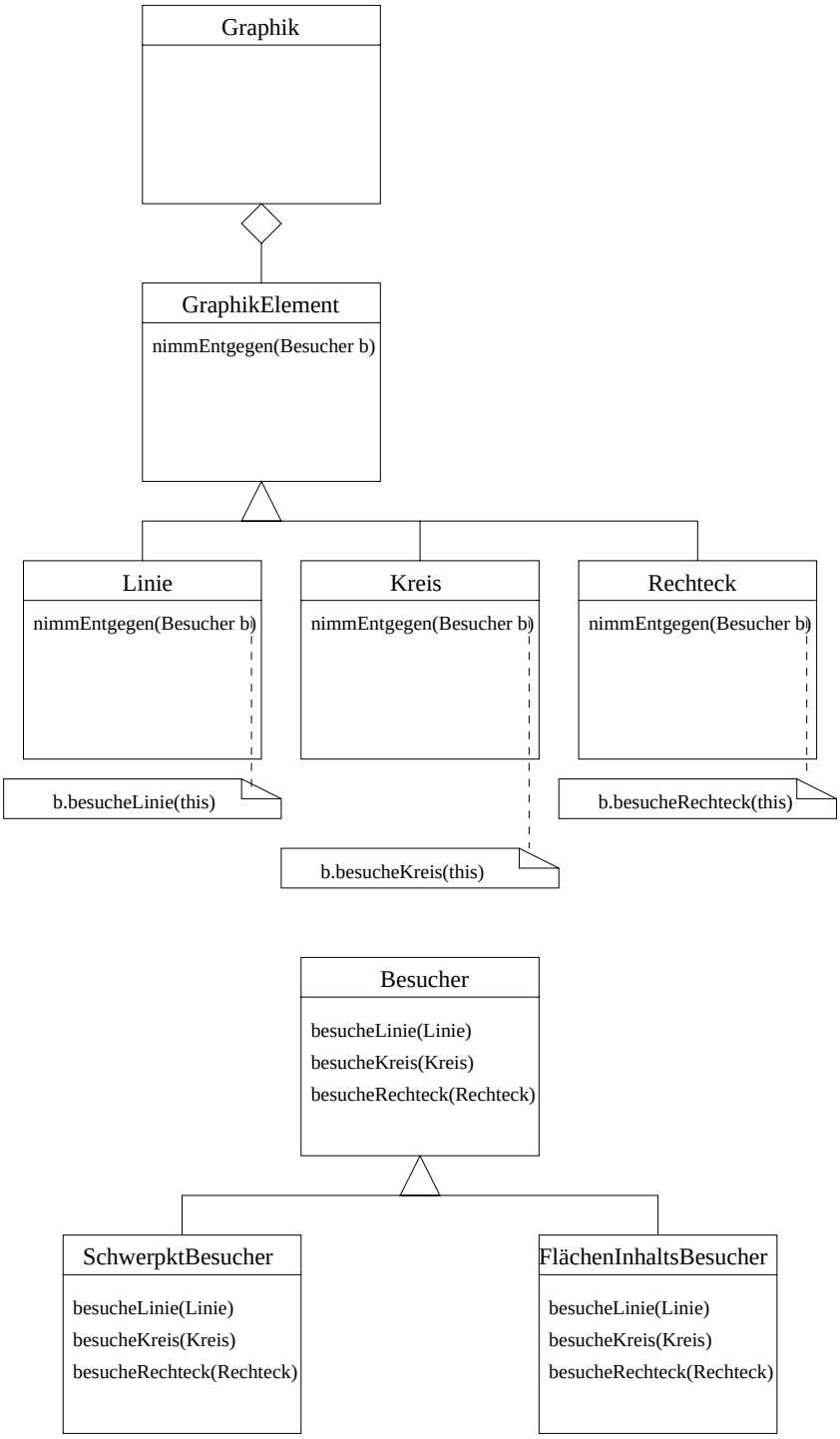
□
```
for (i=0; i<SIZE; i++)
    for (j=0; j<SIZE; j++)
        for (k=0; k<SIZE; k++)
            c[i][k] += a[i][j] * b[j][k];
```

□
```
for (i=0; i<SIZE; i++)
    for (j=0; j<SIZE; j++)
        for (k=0; k<SIZE; k++)
            c[k][j] += a[k][i] * b[i][j];
```

Name: _____ Matrikelnummer: _____

Aufgabe 2 (Entwurfsmuster) **(11 Punkte)**

Gegeben sei folgendes vereinfachtes Klassendiagramm der Graphikeinheit eines CAD-Programms.



Die nächste Version des CAD-Programms soll nun um Operationen auf diesen Graphik-elementen erweitert werden. Mögliche Operationen wären z.B. die Berechnung des Schwerpunktes oder des Flächeninhaltes der gesamten Graphik. Auch soll der Anwender die Möglichkeit erhalten, eigene Operationen definieren und ausführen zu können.
Aus diesen Grund sollen die neuen Operationen mit dem Besucher-Muster verwirklicht werden.

Name: _____ Matrikelnummer: _____

2P
1P pro richtige Antwort

a) Nennen Sie zwei wesentliche Vorteile (ohne Bezug auf die konkrete Aufgabe), die für die Benutzung des Besucher-Musters sprechen.

- *Keine Verschmutzung der Elementklassen mit Operationen.*
- *Dadurch auch: Leichte Erweiterbarkeit.*
- *Zusammengehörige Operationen sind in Besucherobjekten gekapselt.*
- *Parallele Entwicklung verschiedener Operationen möglich.*

3P
1P für „nimmEntgegen“
2P für richtige Implementierung

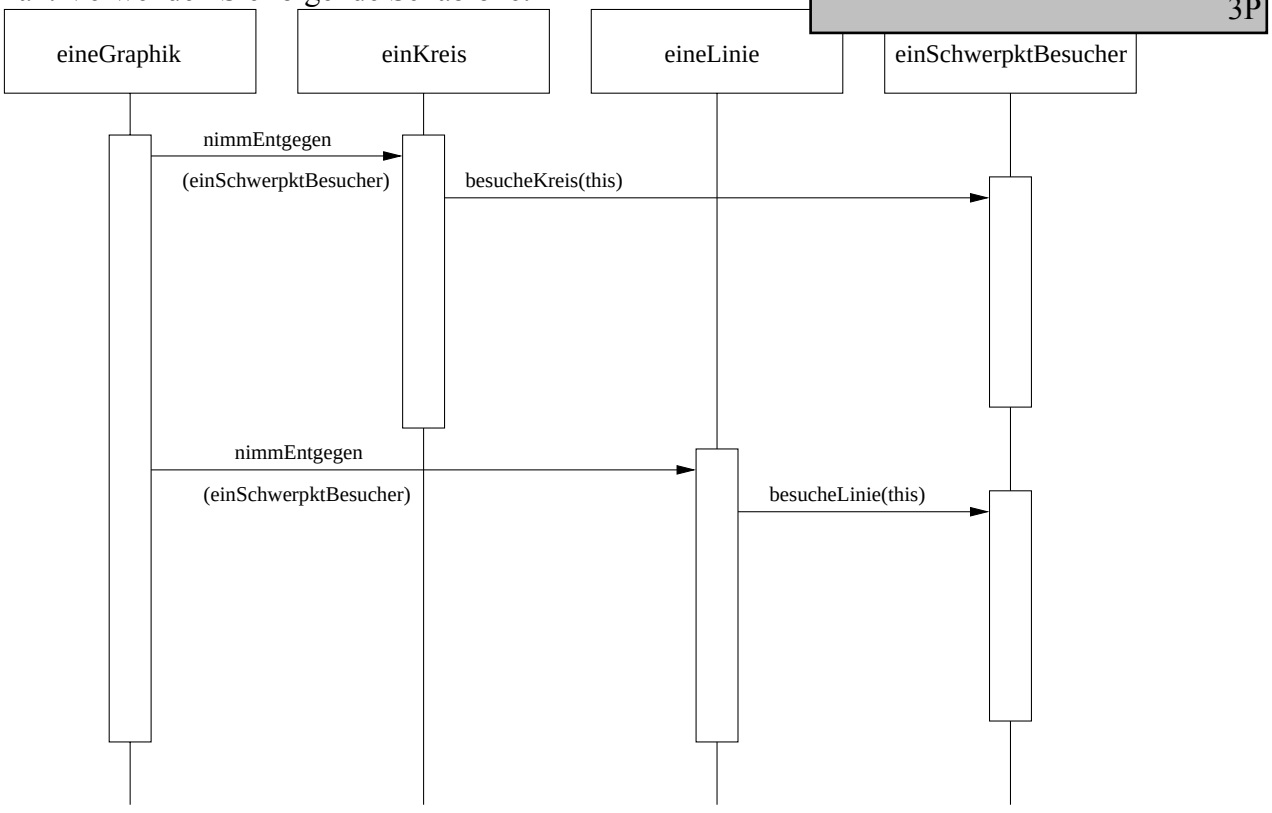
b) Um welche Operation(en) müssen die Klassen *GraphikElement*, *Linie*, *Kreis* und *Rechteck* zur Implementierung des Besucher-Musters erweitert werden? Schreiben Sie die Methodennamen in die Kästen der jeweiligen Klasse und geben Sie die Implementierung jeder neuen Methode an. (Machen Sie sich über bisher nicht eingezeichnete Klassen des Besucher-Musters zunächst noch keine Gedanken.)

3P
1P für richtige Struktur
2P für Operationen

c) Erweitern Sie das Klassendiagramm um den (oben erwähnten,) noch fehlenden Besucher, den *Schwerpunktbesucher* und den *Flächeninhaltsbesucher*. Notieren Sie für jede neue Klasse, welche Besucher-Methoden notwendig sind, aber implementieren Sie diese nicht. (Es ist auch nicht notwendig die *GraphikElemente* um die zur Berechnung notwendigen Attribute oder Methoden zu erweitern.)

3P

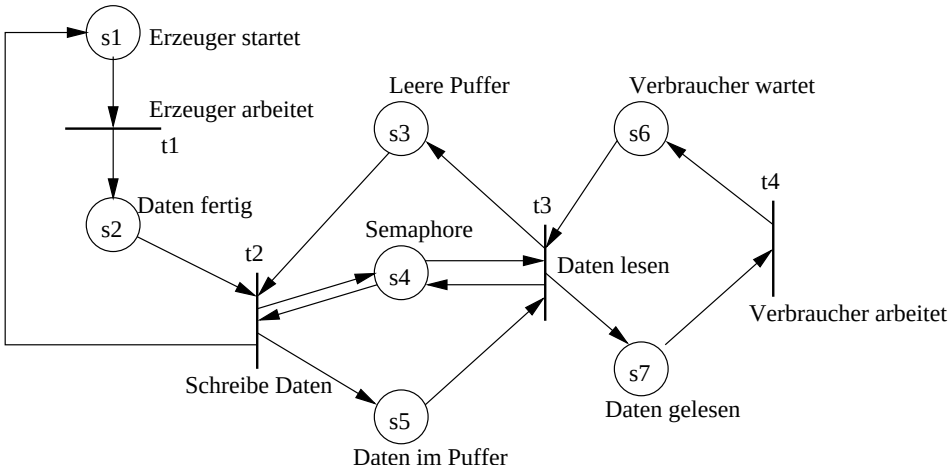
d) Zeichnen Sie ein Sequenzdiagramm für folgendes Szenario: Von einem Graphikobjekt (*eineGraphik)* aus besucht ein Schwerpunktbesucher (*einSchwerpktBesucher*) nacheinander einen Kreis (*einKreis*) und eine Linie (*eineLinie*). Verwenden Sie die Methodennamen aus Teilaufgabe c). Geben Sie die Argumente jedes Methodenaufrufes an. Verwenden Sie folgende Schablone:



Name: _____ Matrikelnummer: _____

Aufgabe 3 (Petrinetz) **(6 Punkte)**

Betrachten Sie folgendes Petrinetz *P*, das einen Erzeuger und einen Verbraucher mit einer Semaphore synchronisiert. Die Anfangsmarkierung sei $M = (1,0,5,1,0,1,0)$.



a) Bestimmen Sie alle elementaren T-Invarianten des Netzes *P*. 2P

d = (1, 1, 1, 1)^T

b) Betrachten Sie folgende Definitionen für S/T-Netze:

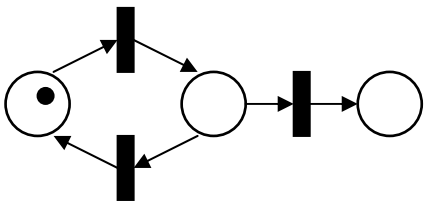
Terminiert: Ein S/T-Netz *P* mit Anfangsmarkierung M terminiert, wenn es keine unendliche Schaltfolge von Transitionen in *P* gibt.

Deadlock-frei: Eine S/T-Netz *P* mit Anfangsmarkierung *M* ist deadlock-frei, wenn jede von *M* aus erreichbare Markierung *M'* eine zulässige Transition besitzt.

b.1) Zeigen Sie: Alle bei Anfangsmarkierung *M* deadlock-freien S/T-Netze terminieren nicht. 2P

Annahme: Es gibt Netz P, deadlock-frei, das terminiert. -> P hat keine unendliche Schaltfolge -> es gibt Markierung, so dass keine Transition schalten kann -> Deadlock. Wiederspruch zur deadlock-freiheit.

b.2) Geben Sie ein Gegenbeispiel zur Umkehrung an, dass jed 2P
Netz deadlock-frei ist.

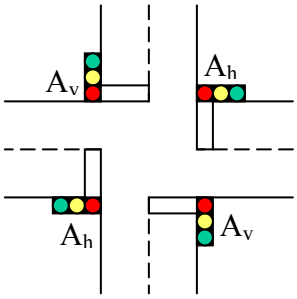


Nicht terminierend, da unendl ich lange Schaltfolge konstruierbar.
Nicht deadlock-frei, da erreichbare Markierung (0, 0, 1) keine weitere Transition besitzt.

Name: _____ Matrikelnummer: _____

Aufgabe 4 (Endliche Automaten) **(12 Punkte)**

In dieser Aufgabe soll ein vereinfachter Automat zur Ampelsteuerung entworfen werden. Betrachten Sie die nebenstehende Kreuzung:
Die mit A_v bezeichneten Ampeln werden jeweils gleich angesteuert. Sie erlauben den Autos in vertikaler Richtung (v) die Durchfahrt. Gleiches gilt für die Ampeln A_h und die horizontale Richtung (h). Vor den Ampeln befinden sich Sensoren in der Straße, die anfahrende Autos an die Ampelsteuerung melden.



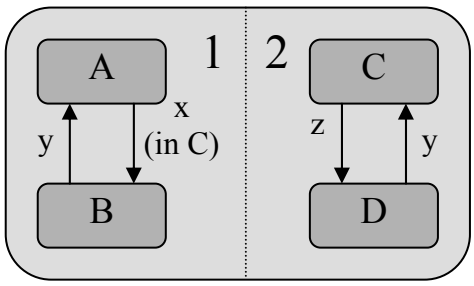
a) Entwerfen Sie auf dem folgenden Blatt einen nebenläufigen Harel-Automaten, der die Steuerung der Ampel-Anlage modelliert und halten Sie sich dabei genau an die folgenden Vorgaben:

Zustände: Von den einzelnen Ampelphasen (rot, rotgelb, grün, gelb) soll abstrahiert werden, so dass die aktuelle Situation (rot, grün) durch zwei Zustände modelliert werden kann. Befindet sich der Automat im Zustand „**V**“, so wird A_v auf grün geschaltet (und A_h auf rot) und die Autos in vertikaler Richtung können fahren. Umgekehrt können in Zustand „**H**“ die Autos in horizontaler Richtung fahren. (Umschaltverzögerungen können vernachlässigt werden.)
Es gibt weitere vier Zustände, die Gün-Anforderungen durch Kontaktschleifen modellieren. Der Zustand „**VA**“ zeigt an, dass Autos in vertikaler Richtung auf die Ampeln A_v zu fahren während diese rot sind. Im Zustand „**kVA**“ möchten keine Autos in vertikaler Richtung die Kreuzung passieren. Analoges gilt für die Zustände „**HA**“, „**kHA**“ und die horizontale Richtung.

Signale: Auf folgende Signale soll die Steuerung reagieren und entsprechende Zustandsänderungen durchführen: In regelmäßigen Abständen (von ca. 2 Minuten) erhält die Steuerung ein Zeitsignal ***„t“***. Fahren Autos über die Kontaktschleifen vor den Ampeln A_v, so erhält die Steuerung das Signal ***„sv“***. Das Signal ***„sh“*** wird entsprechend bei den Ampeln A_h ausgelöst.

Verhalten: Ein Automat schaltet zwischen den Zuständen „**V**“ und „**H**“ zum Zeitpunkt des Signals ***„t“*** um, aber nur wenn Autos in der Querrichtung warten. Dieser Bedarf wird durch zwei parallel dazu laufende Automaten, bestehend aus den Zuständen „**VA**“ und „**kVA**“ bzw. „**HA**“ und „**kHA**“ ermittelt. Das Rücksetzen der Anforderung (d.h. Übergang „**VA**“ in „**kVA**“ bzw. „**HA**“ in „**kHA**“) wird bei dem Signal ***„t“*** durchgeführt, wenn die jeweilige Richtung grün bekommen wird.
Markieren Sie alle Anfangszustände und nehmen Sie dazu eine leere Kreuzung an!

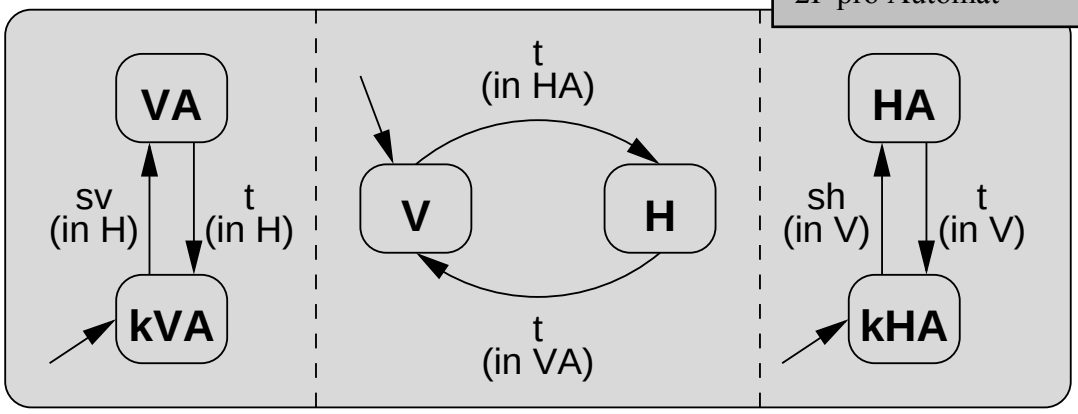
Anmerkung zur Notation:
Bei nebenläufigen Harel-Automaten können Zustandsübergänge von dem aktuellen Zustand eines anderen Automaten abhängig gemacht werden.
Im nebenstehenden Beispiel geht der Automat 1 bei dem Eingabesymbol ***„x“*** nur dann von Zustand „A“ in „B“ über, wenn sich Automat 2 in Zustand „C“ befindet Nutzen Sie diese Notation zur Lösung des Aufgabenteils a) !



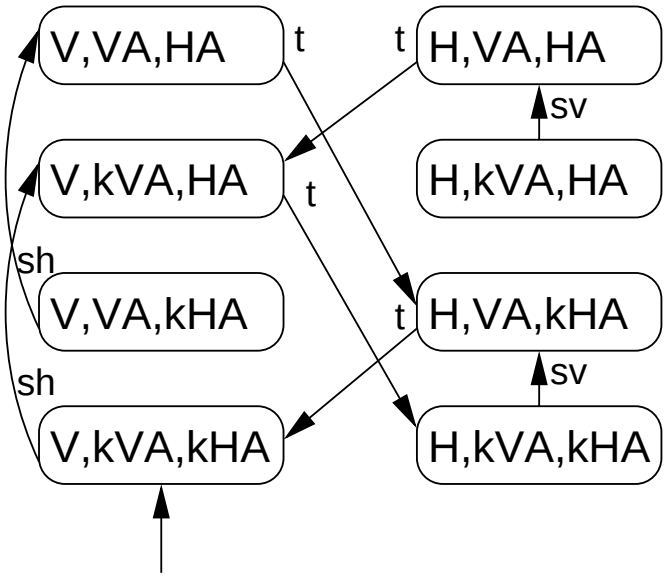
Name: _____ Matrikelnummer: _____

Lösung a) 6P

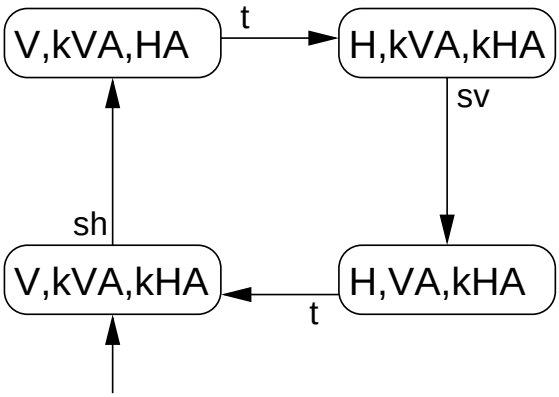
~2P pro Automat



b) Wandeln Sie Ihren in Aufgabenteil a) konstruierten nebenläufigen Harel-Automaten in einen Mealy-Automaten um. 6P



vereinfacht:



Name: _____ Matrikelnummer: _____

Diese Aufgabe ist von Infowirten nicht zu bearbeiten!

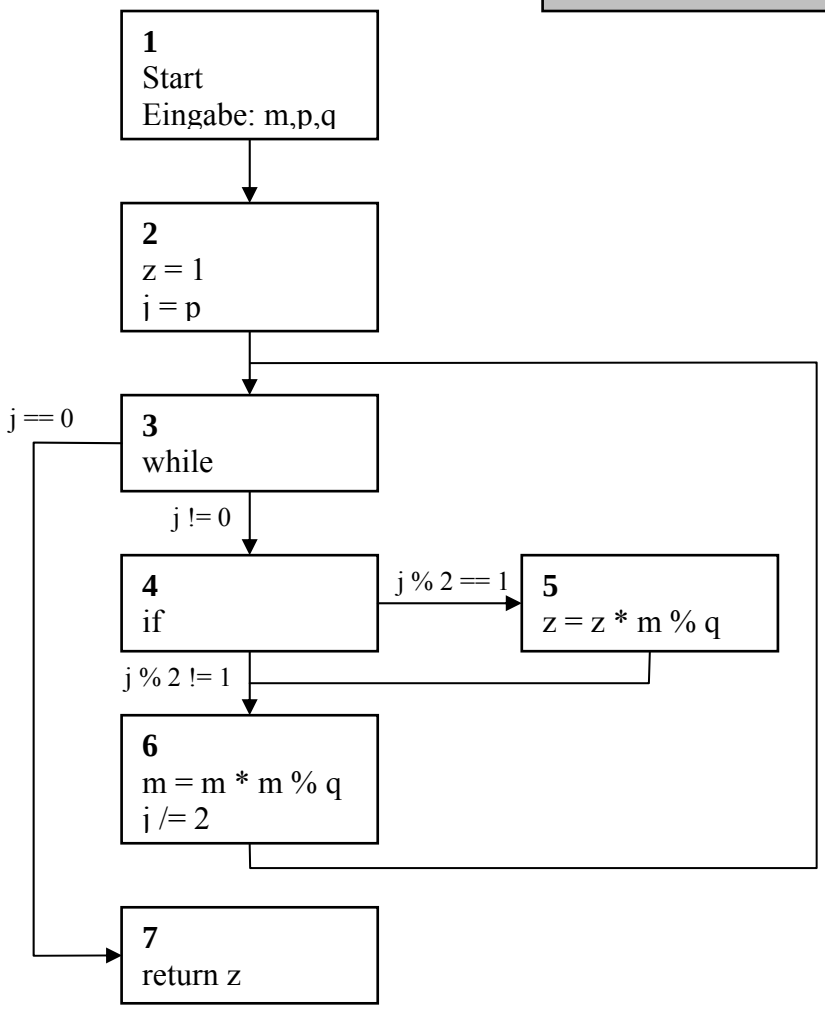
Aufgabe 5 (Datenflussorientierter Strukturtest) **(16 Punkte)**

Gegeben sei folgende Java Methode:

```
int f(int m, int p, int q) {
     int z = 1;
     int j = p;
     while (j != 0) {
          if (j % 2 == 1)
               z = (z * m) % q;
          m = (m * m) % q;
          j /= 2;
     }
     return z;
}
```

a) Vervollständigen Sie die untenstehende Schablone des Kontrollflussgraphen G der Funktion *f*. Verwenden Sie dabei für die If- und While-Anweisungen leere Knoten und numerieren Sie Ihre Knoten in der Reihenfolge in der sie im Quelltext auftreten.

G: 3P



Name: _____ Matrikelnummer: _____

b) Welche Funktion berechnet das Programm?

1P

$$f(m, p, q) = (m \wedge p) \bmod q$$

c) Tragen Sie in die untenstehenden Tabellen nach dem Definitions-Nutzungs-Verfahren alle Definitionen, r-Nutzungen und p-Nutzungen ein.

3P
linke Tabelle: 2P
rechte Tabelle: 1P

Knoten K	def(k)	r-Nutzung(K)
1	{m, p, q}	{}
2	{z, j}	{p}
3	{}	{}
4	{}	{}
5	{z}	{z, m, q}
6	{m, j}	{m, j, q}
7	{}	{z}

Kante K	p-Nutzung(K)
(1,2)	{}
(2,3)	{}
(3,4)	{j}
(3,7)	{j}
(4,5)	{j}
(4,6)	{j}
(5,6)	{}
(6,3)	{}

d) Geben Sie unter Verwendung von c) alle drn- und dpn-Mengen an.

4P
linke Spalte: 3P
rechte Spalte: 1P

drn(m,1) = {5,6}	dpn(m,1) = {}
drn(p,1) = {2}	dpn(p,1) = {}
drn(q,1) = {5,6}	dpn(q,1) = {}
drn(z,2) = {5,7}	dpn(z,2) = {}
drn(j,2) = {6}	dpn(j,2) = {(3,4), (3,7), (4,5),
drn(z,5) = {5,7}	dpn(z,5) = {}
drn(m,6) = {5,6}	dpn(m,6) = {}
drn(j,6) = {6}	dpn(j,6) = {(3,4), (3,7), (4,5),

e) Geben Sie für die folgenden Parameter (m, p, q) jeweils an, welcher Pfad im Programm durchlaufen wird und welche Kriterien der Pfad erfüllt. (Hinweis: Machen Sie sich klar, wie der Programmablauf von den Eingabeparametern abhängt.)

5P
1P / komplett richtige Zeile

(m, p, q)	Pfad	1	2	3	4	5
(2, 0, -1)	1, 2, 3, 7					
(3, 3, 10)	1, 2, 3, 4, 5, 6, 3, 4, 5, 6, 3, 7	x		x		
(4, 1, 10)	1, 2, 3, 4, 5, 6, 3, 7	x				
(2, 5, 16)	1, 2, 3, 4, 5, 6, 3, 4, 6, 3, 4, 5, 6, 3, 7	x	x	x		
(4, 6, 0)	1, 2, 3, 4, 6 (Division durch 0)					

Kriterien: 1: alle Knoten 2: alle Kanten 3: alle Definitionen 4: alle p-Nutzungen 5: alle r-Nutzungen