Klausur Softwaretechnik

8. April 2002

Name: _____

Vorname: _____

Matrikelnummer: _____

Aufg.0	Aufg.1	Aufg.2	Aufg.3	Aufg.4	Aufg.5	Σ	Note
/1	/18	/13	/10	/12	/6	/60	

Die Klausur besteht aus 15 Seiten und ist geheftet abzugeben.
Für die Vollständigkeit nicht gehefteter Klausuren wird keine Verantwortung übernommen.

Informationswirte brauchen Aufgabe 5 und Aufgabenteile g) - 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 (18 Punkte)

Für InfoWirte: (12,5 Punkte)

Beantworten Sie folgende Fragen.
Falsche Kreuze geben negative Punkte, fehlende Kreuze, sowie fehlende oder falsche Freitext-Antworten bewirken nichts. Weniger als 0 Punkte können Sie mit dieser Aufgabe insgesamt nicht erreichen.

a) Nennen Sie drei Konfliktlösungsstrategien, die bei vorwärtsverketteten Regelsystemen zum Einsatz kommen.

1. _____

2. _____

3. _____

b) Betrachten Sie folgende Klassendefinitionen:

```
public class A {
  public int test(B b) {
    return b.test();
  }
}

public class B {
  private A a;
  public int test() {
    return 42;
  }
}
```

```
public class C extends B {
  public int test() {
    return 2 * super.test();
  }
}

public class D {
  public void test() {
    C c = new C();
    A a = new A();
    a.test(c);
  }
}
```

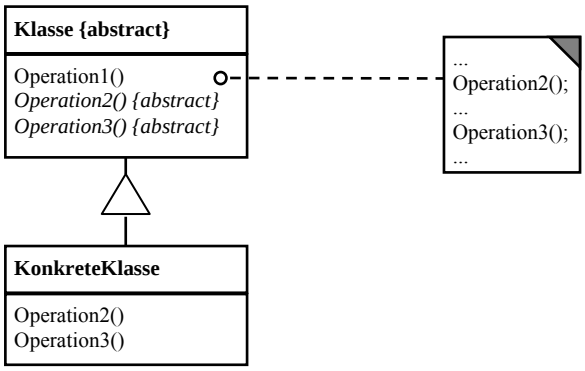
Welche der folgenden Aussagen trifft zu?

Richtig/Falsch

□/□ A benutzt B
□/□ D benutzt B
□/□ C benutzt B
□/□ B benutzt A

c) Welches aus der Vorlesung bekannte Entwurfsmuster ist in der untenstehenden Abbildung zu sehen?

_____



Wie werden die Operationen 1, 2 und 3 genannt?

Operation1: _____

Operation2,3: _____

d) Betrachten Sie folgenden Ausschnitt aus einem Java-Programm:

```
...
if (x>0 & y!=0)
   y = 0;
...
```

Kreuzen Sie in der untenstehenden Tabelle an, welche Kriterien des kontrollflussorientierten Strukturtests durch die Testdaten in dem angegebenen Programmstück jeweils erfüllt werden. (Hinweis: Der Operator **&** ist das logische *Und* ohne Kurzauswertung.)
Punktevergabe: In diesem Aufgabenteil gibt es nur Punkte für komplett richtige Zeilen, falsche Kreuze geben hier keine Abzüge!

Testdaten: (x,y)	1	2	3	4	5
(1, 1)	□	□	□	□	□
(3, 0), (0, 3)	□	□	□	□	□
(1, 0), (5, 5), (-1, -3)	□	□	□	□	□
(2, 0), (3, 5), (0, 0), (-6, 6)	□	□	□	□	□

1: Anweisungsüberdeckung
2: Zweigüberdeckung
3: Pfadüberdeckung
4: einfache Bedingungsüberdeckung
5: mehrfache Bedingungserfassung

e) Welche der folgenden Aussagen über Pfad-Selektions-Kriterien beim Datenflussorientierten Strukturtest sind richtig?

Richtig/Falsch

□/□ „*alle p-Nutzungen*“ schließt „*alle Kanten*“ echt ein.
□/□ „*alle r-Nutzungen*“ schließt „*alle Knoten*“ echt ein.
□/□ „*alle Nutzungen*“ schließt „*alle Pfade*“ echt ein.
□/□ „*alle Nutzungen*“ schließt „*alle Definitionen*“ echt ein.

f) Erstellen Sie aus folgendem Datenwörterbuch ein Syntax-Diagramm:

TeilnehmerDatei = {Teilnehmer}
Teilnehmer = Name + Matrikelnummer + (Geburtsdatum)
+ [Infowirt | Informatiker]

Die folgenden Aufgabenteile müssen von Informationswirten nicht bearbeitet werden!

g) Welche der folgenden Aussagen über „*eXtreme Programming*“ sind richtig?

Richtig/Falsch
- □/□ Die Entwickler entscheiden, welche Funktionalität innerhalb einer Iteration implementiert wird.
- □/□ Eine Iteration sollte nur eine oder wenige Wochen dauern.
- □/□ Während einer Iteration spezifiziert nicht nur jeder Entwickler ausführbare Testfälle, sondern auch der Kunde.
- □/□ Inspektionen finden im wöchentlichen Rhythmus statt und sind Teil der normalen Arbeitszeit.

h) Welche der folgenden Aussagen über die für XML verwendeten DTDs treffen zu?

Richtig/Falsch
- □/□ Man nennt ein Dokument genau dann „wohlgeformt“ (well-formed), wenn es den Regeln einer DTD entspricht.
- □/□ Eine DTD bildet die Schnittstelle zwischen verschiedenen Entwicklerteams.
- □/□ Eine DTD gibt an, welche Attribute und Unterelemente ein Element haben kann.
- □/□ Eine DTD ist selber in XML-Notation formuliert.

i) Finden und korrigieren Sie drei Fehler im folgenden XML-Dokument.

```
<?xml version="1.0"?>
<klausur name=Softwaretechnik>
  <termin tag="8" monat="April" jahr="2002">
     <uhrzeit dauer="1h">14 Uhr
  </termin></uhrzeit>
  <ort>
     <hoersaal>HS am Forum
  </ort>
</klausur>
```

Name: _____ Matrikelnummer: _____

Aufgabe 2 (Objektorientierter Entwurf, Entwurfsmuster) **(13 Punkte)**

In der Vorlesung wurde der modulare Entwurf eines Programmes zur Erzeugung eines *KWIC-Indexes* vorgestellt. In dieser Aufgabe sollen Sie nach den unten angegebenen Gesichtspunkten einen objektorientierten Entwurf zur Berechnung des *KWIC-Indexes* verbessern.
Zur Erinnerung: Beim *KWIC-Index* handelt es sich um einen permutierten Index (KWIC = **K**ey**W**ord **I**n **C**ontext). Er wird aus einer Folge von Zeilen (jeweils bestehend aus einer Folge von Worten) erzeugt. Zu jeder Zeile werden alle zirkulären Verschiebungen durch wiederholtes Entfernen des ersten Wortes und Anhängen am Ende der Zeile gebildet. Das Ergebnis (alle Verschiebungen aller Zeilen) wird anschließend sortiert.

Beispiel:	Eingabe:	„Salt Lake City“
	Zirkuläre Verschiebungen:	„Salt Lake City“, „Lake City Salt“, „City Salt Lake“
	KWIC-Index (sortiert):	„City Salt Lake“, „Lake City Salt“, „Salt Lake City“

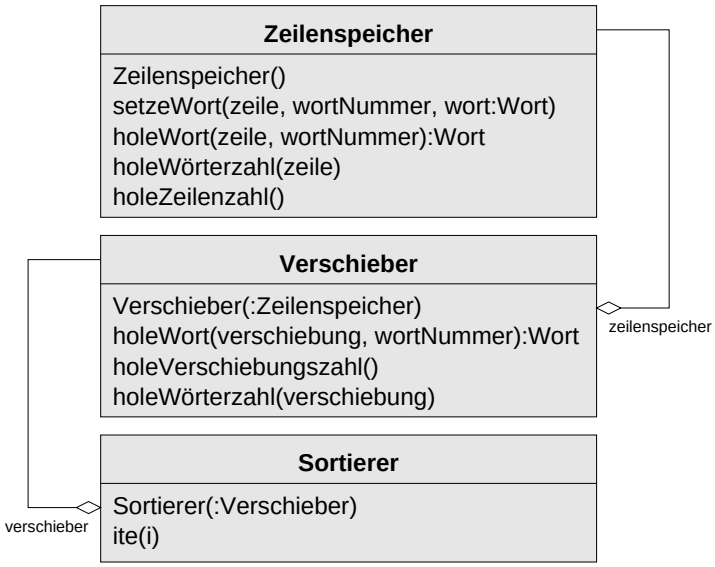
Aus dem modularen Entwurf sei der auf der folgenden Seite dargestellte objektorientierte Entwurf entstanden.
Erläuterung: Die Klasse *Zeilenspeicher* speichert die Eingabezeilen und erlaubt den Zugriff auf diese Daten über die Methoden *setzeWort*, *holeWort*, *holeWörterzahl* und *holeZeilenzahl*. Der *Zeilenspeicher* wird zu Beginn einmal beschrieben und dann nicht mehr verändert. Die Klasse *Wort* ist nicht angegeben: Sie dient der Speicherung von einzelnen Worten und entspricht in der Funktionalität ungefähr der von Java bekannten Klasse *java.lang.String*.
Der *Verschieber* greift auf ein Objekt vom Typ *Zeilenspeicher* zu und berechnet alle zirkulären Verschiebungen der dort gespeicherten Zeilen. Die Methode *holeVerschiebungszahl* gibt die Gesamtzahl der so entstehenden Verschiebungen zurück. Die Anzahl der Wörter in einer Verschiebung kann über *holeWörterzahl* bestimmt werden und auf die einzelnen Verschiebungen kann mit Hilfe einer dem *Zeilenspeicher* ähnlichen Schnittstelle (*holeWort*) zugegriffen werden. Der *Sortierer* wird mit einem *Verschieber* initialisiert und bietet die Permutationsfunktion *ite(i)*, welche die Nummer der Verschiebung liefert, die an der i´ten Stelle der Sortierung steht.

Problem: Dieser Entwurf ist noch stark von dem vorgegebenen modularen Entwurf geprägt. Insbesondere ist für die Abfrage des Ergebnisses die Kenntnis von sowohl der *Sortierer*- wie auch der *Verschieber*-Klasse notwendig.

Aufgabe: Hauptziel des neuen Entwurfs wird sein, allen Klassen die gleiche Schnittstelle (*KwicModul*) zu geben. Beispielsweise soll das Auslesen der Ergebnisse aus dem *Sortierer* genau wie die Abfrage der Daten aus dem *Zeilenspeicher* ablaufen.
Der OO-Entwurf soll das Entwurfsmuster Dekorierer (Stellvertreter) verwenden und die Klasse *Zeilenspeicher* um die zusätzliche Funktionalität erweitern. Folgen Sie den folgenden Arbeitsschritten und ergänzen dabei das Klassendiagramm auf Seite 9.

a) Entwerfen Sie die abstrakte Schnittstelle *KwicModul* als Oberklasse der Klasse *Zeilenspeicher*. Diese Schnittstelle muss alle Operationen enthalten, die zum *lesenden* Zugriff auf die Daten im *Zeilenspeicher* notwendig sind, aber nicht mehr.

b) Fügen Sie dem Diagramm die Klasse *ZSDekorierer* hinzu, von welcher die Klassen, die die Funktionalität des *Zeilenspeicher*s erweitern, abgeleitet werden sollen. Geben Sie alle von der Klasse zu implementierenden Operationen an. Zeichnen Sie eventuell vorhandene Vererbungen und Assoziationen ein und benennen Sie die Assoziation(en).



c) Implementieren Sie alle Operationen (außer dem Konstruktur) der Klasse *ZSDekorierer* in einer Java-ähnlichen Syntax. (Führen Sie die Implementierungen im Klassendiagramm aus.)

d) Entwerfen Sie die konkreten Dekorierer (*Sortierer* und *Verschieber*) und fügen Sie sie dem Klassendiagramm hinzu. Geben Sie dabei für jeden Dekorierer die Signaturen der von ihm zu implementierenden Operationen an.

e) Implementieren Sie in Java die Methode **`erzeugeKwicIndex`**. Sie erhält als Parameter ein Objekt vom Typ *Zeilenspeicher* **`zs`** und liefert als Ergebnis ein *KwicModul*, welches den KWIC-Index aus **`zs`** berechnet.
Benutzen Sie die folgende Schablone:

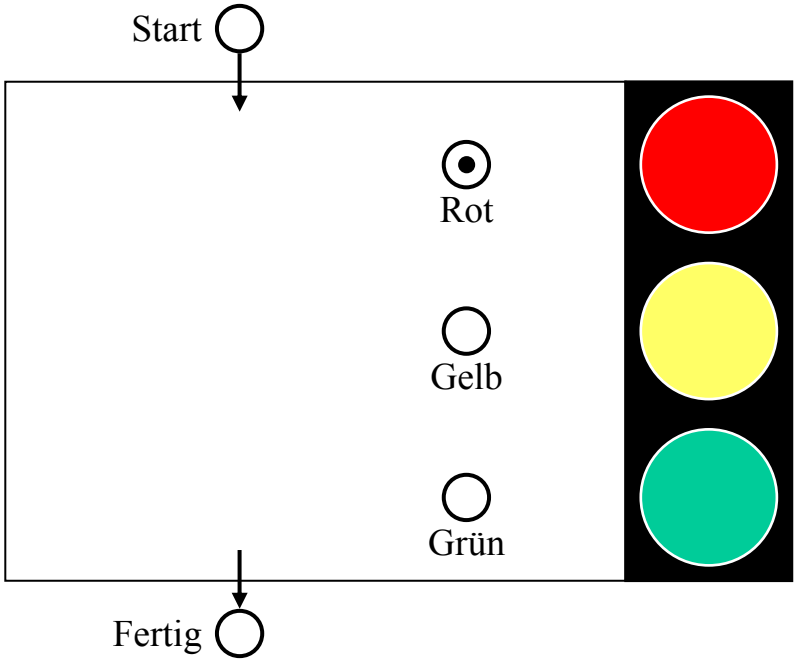
```
public KwicModul erzeugeKwicIndex(Zeilenspeicher zs) {

    _____

    _____

    return _____

}
```

Name: _____ Matrikelnummer: _____

Zeilenspeicher
Zeilenspeicher() setzeWort(zeile, wortNummer, wort:Wort) holeWort(zeile, wortNummer):Wort holeWörterzahl(zeile) holeZeilenzahl()

Aufgabe 3 (Petri-Netz) **(10 Punkte)**

In dieser Aufgabe soll eine Ampelanlage auf verschiedenen Ebenen mit Hilfe von Petri-Netzen modelliert werden. Die Grundidee dabei ist, dass eine einzelne Ampel eigenständig den Zyklus durchläuft und sich zu bestimmten Zeitpunkten mit anderen Ampeln synchronisiert.
Die folgende Abbildung zeigt ein einzelnes Ampel-Modul:



Das Modul ist eine Verfeinerung der Transition „Schalte eine Ampel durch einen Zyklus“. Als Schnittstelle besitzt das Modul die Eingabestelle *Start* und die Ausgabestelle *Fertig*.
Die drei Stellen *Rot*, *Gelb* und *Grün* bilden die interne Schnittstelle zu den Lampen: Enthalten sie eine Marke, so leuchtet die entsprechende Lampe auf. In der Anfangsmarkierung ist die Ampel auf Rot geschaltet, daher befindet sich eine Markierung in der Stelle *Rot*.
Sobald die Stelle *Start* eine Markierung enthält, kann der (bekannte) Zyklus durchlaufen werden: **Rot-Gelb**, **Grün**, **Gelb** und anschließend wieder **Rot**. Hat die Ampel den Durchlauf absolviert, so stoppt sie und legt eine Markierung in die Stelle *Fertig*. Die übrigen Stellen müssen wieder so belegt sein wie vor dem Durchlauf. Der nächste Zyklus beginnt erst dann, wenn er durch Ablage einer Markierung in *Start* freigegeben wird.

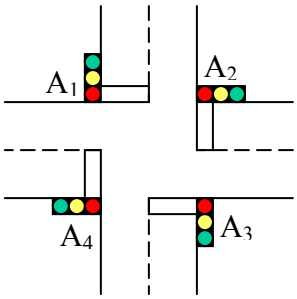
a) Ergänzen Sie das Modul um die notwendigen Stellen und Transitionen, so dass es gemäß der obigen Beschreibung funktioniert und sich in der Anfangsmarkierung befindet. Lassen Sie zunächst Zeitverzögerungen außer Acht.

b) Machen Sie aus ihren Petri-Netz ein *zeitbehaftetes*, indem Sie einzelne Stellen mit der Mindestverweildauer der Marken nach folgender Tabelle annotieren. (Die Stellen *Start* und *Fertig* liegen außerhalb des Moduls und dürfen daher nicht annotiert werden.)

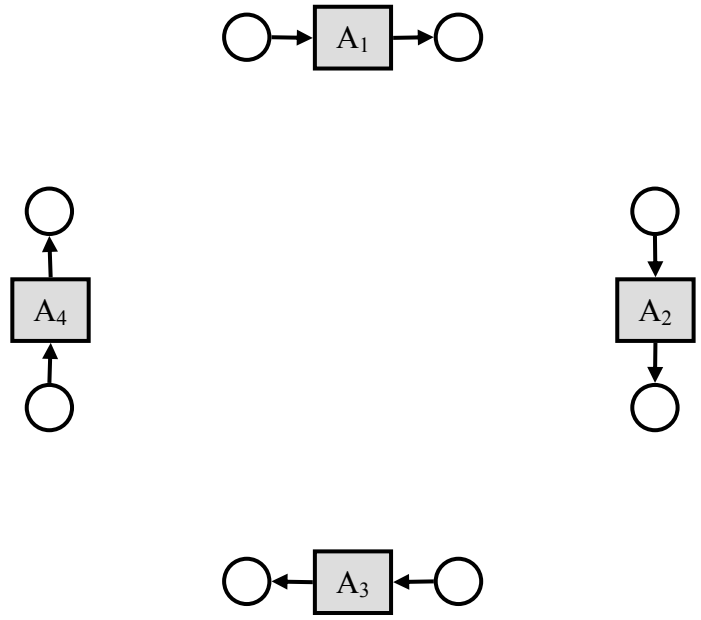
Phase	Mindestdauer
Rot	15 sec
Rot-Gelb	2 sec
Grün	15 sec
Gelb	3 sec

Name: _____ Matrikelnummer: _____

Mit Hilfe dieser Ampelmodule soll nun eine gesamte Kreuzung, wie in der nebenstehenden Abbildung beschaltet werden. Die einzelnen Ampeln A_1, A_2, A_3 und A_4 sind dabei eigenständig und funktionieren nach dem beschriebenen Prinzip.
Zur Synchronisation werden die Schnittstellen *Start* und *Fertig* benutzt, so dass jeweils A_1 und A_3 respektive A_2 und A_4 gleichzeitig den Ampelzyklus durchlaufen, während die anderen Ampeln auf Rot stehen.



c) Entwerfen Sie mit den in Aufgabenteil a) und b) entworfenen und unten vorgegebenen Ampelmodulen A_1, A_2, A_3 und A_4 ein Petri-Netz, welches die Kreuzung richtig beschaltet. (Anfangsmarkierung nicht vergessen!)



d) Welches ungewöhnliche Verhalten kann diese Ampelschaltung zeigen?

_____

_____

_____

Welche Eigenschaft der hier verwendeten Petri-Netze (genauer: ihrer Transitionen) ist die Ursache dafür?

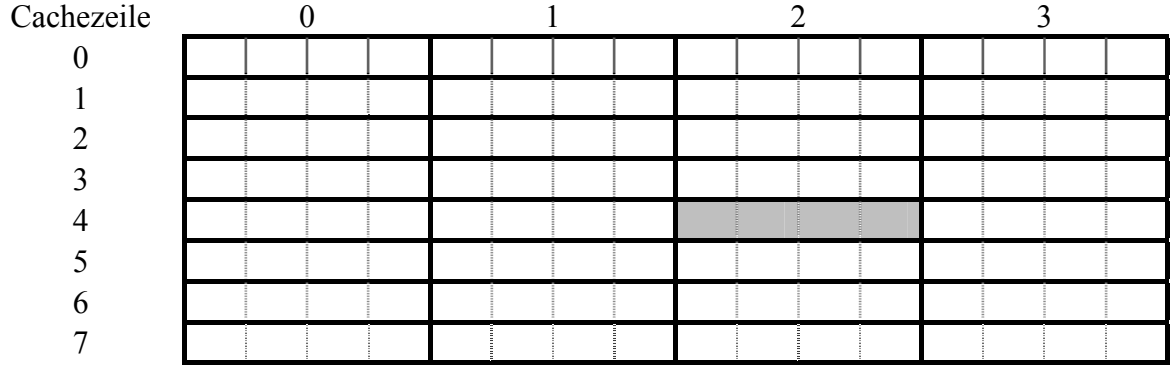
_____

_____

_____

Aufgabe 4 (Cache, Optimierung) **(12 Punkte)**

Gegeben sei ein direkt abgebildeter (Assoziativität=1) Cache. Er sei 128 Byte groß und habe 8 Cachezeilen. Jede Cachezeile besteht aus 16 Byte. Der Datentyp **`real`** besteht aus 4 Byte. Somit passen in jede Cachezeile immer genau 4 Datenelemente des Typs real. Die folgende Zeichnung veranschaulicht noch einmal den Aufbau des Caches.



Die einzelnen Kästchen entsprechen einem Byte; 4 können ein Element des Datentyps **`real`** aufnehmen, wie z.B. der grau markierte Bereich in Cachezeile 4 und Position 2.

Betrachten Sie nun folgende Programmschleife in C:

P

```
     ...
11   for ( j = 1; j<=100; j++ ) {
12     for ( i=0; i<15; i++ ) {
13       c[i+7] = ( a[i] + a[i+1] ) / 2;
14       b[i] = b[i] + c[i+7];
15     }
16   }
     ...
```

Die Felder **`a`**, **`b`**, **`c`** haben den Elementtyp **`real`**. Alle anderen Variablen sind vom Typ **`int`**. Die Programmschleife berechnet zunächst den Mittelwert zweier Elemente von **`a`** (Zeile **13**). Diese Mittelwerte werden dann auf die Elemente des Feldes **`b`** aufaddiert (Zeile **14**).
Die Elemente **`c[i+7]`** (für 0<= **`i`** <15) werden später im Programm nicht mehr gebraucht.

Für die Anfangsadressen der Felder im Hauptspeicher gilt
adr(**`a[0]`**) mod 128 = 0
adr(**`b[0]`**) mod 128 = 64
adr(**`c[0]`**) mod 128 = 48.

Nehmen Sie für die folgenden Aufgaben an, dass alle skalaren Variablen in Registern des Prozessors liegen und dass die Elemente der Felder **`a`**, **`b`** und **`c`** über den Cache in den Prozessor geladen werden.

Name: _____ Matrikelnummer: _____

a) Markieren Sie jeweils in den entsprechenden Cacheabbildungen die Cache-Positionen der Feldelemente von **a**, **b** und **c**, die bei der ersten Iteration (**i**=0) der **i**-Schleife zugegriffen werden mit einem O (Kreis).

b) Markieren Sie nun den gesamten Cache-Bereich, der von den Feldern **a**, **b** und **c** beim ersten kompletten Durchlauf der **j**-Schleife (**j**=1; 0<=**i**<15) benutzt wird.

Cacheabbildung von **a**

Cachezeile	0	1	2	3
0				
1				
2				
3				
4				
5				
6				
7				

Cacheabbildung von **b**

Cachezeile	0	1	2	3
0				
1				
2				
3				
4				
5				
6				
7				

Cacheabbildung von **c**

Cachezeile	0	1	2	3
0				
1				
2				
3				
4				
5				
6				
7				

c) Tragen Sie in der folgenden Tabelle die Cachebelegung nach der ersten Iteration der **j**-Schleife ein. (Es genügt, die letzte Spalte auszufüllen.)

Cachezeile	0	1	2	3	Belegt durch Elemente aus Feld (**a**/**b**/**c**)
0					
1					
2					
3					
4					
5					
6					
7					

d) Wir betrachten nun eine der Iterationen 2 bis 100 der **j**-Schleife: Wie viele Cachezeilen müssen während der 15 Iterationen der **i**-Schleife für die jeweiligen Felder neu geladen werden?

Zeile **13**:

Feld **a**: _____ Feld **c**: _____

Zeile **14**: (Es wird zuerst auf das Feld **c** zugegriffen.)

Feld **c**: _____ Feld **b**: _____

e) Durch welche kleine Modifikation der Zeilen **13** und **14** kann die Anzahl der neu zu ladenden Cachezeilen auf 0 reduziert werden? (Hinweis: Beachten Sie die anfangs beschriebenen Randbedingungen!)
Erklären Sie kurz Ihr Vorgehen und notieren Sie unten die beiden neuen Zeilen.

Erklärung:

_____

_____

_____

_____

_____

Zeile 13': _____

Zeile 14': _____

Name: _____ Matrikelnummer: _____

Diese Aufgabe muss von Informationswirte nicht bearbeitet werden!

Aufgabe 5 (Aufwandsschätzung) **(6 Punkte)**

Es soll das Produkt **P** entwickelt werden.
Sie sind für die Aufwandsschätzung zuständig. Die Projektleitung erwartet von Ihnen folgende Kenngrößen Ihrer Schätzung: *LOC*, *Entwicklungsaufwand*, *optimale Entwicklungsdauer*, die *durchschnittliche Größe des Entwicklerteams* und eine erste *Kostenschätzung*.

Beantworten Sie dazu folgende Fragen. Bei Berechnungen interessiert uns nur der Lösungsweg, deshalb sind viele Kenngrößen nur als Variablen angegeben.

Schätzen Sie zunächst die Größe von **P** mit Hilfe der parametrischen Gleichungen. Ihre Firma hat dazu aus Ihren Projekthistorien folgende Kennziffern ermittelt:

Kategorie	Faktor	Teilgröße in LOC für **P**
Steuerprogramm	2,0	$\mathbf{LOC_S}$
Ein-/Ausgabe	1,2	$\mathbf{LOC_{E/A}}$
Datenverwaltung	1,0	$\mathbf{LOC_D}$
Algorithmen	2,5	$\mathbf{LOC_A}$

a) Stellen Sie die parametrische Gleichung auf, mit der Sie die *bewerteten LOC* berechnen.

$\mathbf{LOC_{bewertet}}$ = _____

b) Wie viele Quellkodezeilen liefert ein Softwareentwickler im Monat (Faustregel nach Balzert)?

z = _____ LOC

c) Wie berechnen Sie daraus den Entwicklungsaufwand für **P** in Mann-Monaten?

E = _____

d) Wie berechnen Sie daraus nach *CoCoMo* die optimale Entwicklungsdauer in Monaten. Die Art des Systems **P** sei mit den Konstanten **a** und **s** modelliert.

T = _____

e) Berechnen Sie die durchschnittliche Größe **G** des Entwicklerteams.

G = _____

f) Mit welchen Entwicklungskosten muss nun Ihre Firma rechnen, wenn ein Entwickler 70.000 € pro Jahr kostet.

K = _____