

Lösungshinweise Klausur INFORMATIK - I

24. Februar 2006

*Kleben Sie hier das Etikett mit Ihrem
Namen und Ihrer Matrikelnummer auf.*

Hinweise

- **Kleben Sie das Etikett mit Ihrem Namen und Ihrer Matrikelnummer auf das Titelblatt!**
- **Versehen Sie alle Antwortblätter mit Ihrem Namen und Ihrer Matrikelnummer!**
- Schreiben Sie die Lösungen nur auf das entsprechende Aufgabenblatt. Wo angegeben, verwenden Sie bitte die bereits vorgegebenen Skizzen. Sie können die Antwortblätter auf beiden Seiten beschreiben, müssen dies aber auf dem Aufgabenblatt vermerken. Sollte Ihnen wider Erwarten der Platz auf einem Blatt nicht ausreichen, so können Sie durch Handzeichen von der Klausuraufsicht weitere Blätter erhalten, **die Sie wiederum mit Ihrem Namen und Ihrer Matrikelnummer kennzeichnen.**
- Insgesamt können **60 Punkte** erreicht werden!
- **Geben Sie alle Blätter incl. des Titelblatts wieder ab**, da nachgereichte Blätter nicht korrigiert werden. Heften Sie vor der Abgabe die Blätter mit der Büroklammer zusammen.

Viel Erfolg!

Aufgabe:	1	2	3	4	5	6	7	Σ
Punkte:								
Note:								

1 Textersetzung (10 Punkte)

- a) Entwerfen Sie ein Semi-Thue-System mit möglichst wenig Regeln, das zu einer gegebenen Bitfolge das Einerkomplement bestimmt. Dazu muss jede 1 in eine 0 und jede 0 in eine 1 umgewandelt werden. Die Bitfolge als Eingabewort beginnt und endet jeweils mit α . Beide α dürfen im Ergebnis nicht mehr vorkommen. **(4 Punkte)**

Beispiel: $\alpha 001\alpha$ wird abgeleitet zu 110

Lösung

$\alpha 0 \mapsto 1\alpha$ (1)
 $\alpha 1 \mapsto 0\alpha$ (2)
 $\alpha\alpha \mapsto \varepsilon$ (3)

- b) Geben Sie eine Ableitung für das Wort $\alpha 001\alpha$ und bei jedem Ableitungsschritt die verwendete Regel an. **(1 Punkt)**

$\alpha 001\alpha \xrightarrow{1} 1\alpha 01\alpha \xrightarrow{1} 11\alpha 1\alpha \xrightarrow{2} 110\alpha\alpha \xrightarrow{3} 110$

- c) Entwerfen Sie ein Semi-Thue-System, das mit möglichst wenig Regeln zu einer gegebenen Bitfolge die Differenz der Anzahl der Zeichen 1 und der Anzahl der Zeichen 0 folgendermaßen ausgibt: **(4 Punkte)**

$n - m$ mal das Zeichen 0 falls $n > m$
 $m - n$ mal das Zeichen 1 falls $m > n$
 ε sonst

n ist dabei die Anzahl der Zeichen 1, m die Anzahl der Zeichen 0 im Eingabewort.

Beispiele:

001010 wird abgeleitet zu 00

101000 wird abgeleitet zu 00

111010 wird abgeleitet zu 11

Lösung

$10 \mapsto \varepsilon$ (1)
 $01 \mapsto \varepsilon$ (2)

- d) Geben Sie eine Ableitung für das Wort 01101 und bei jedem Ableitungsschritt die verwendete Regel an. **(1 Punkt)**

$01101 \xrightarrow{2} 101 \xrightarrow{1} 1$

2 Akzeptor und Grammatik (10 Punkte)

Gegeben sei die Sprache L über $\Sigma = \{1, 0\}$, bei der für alle Worte $w \in L$ gilt:

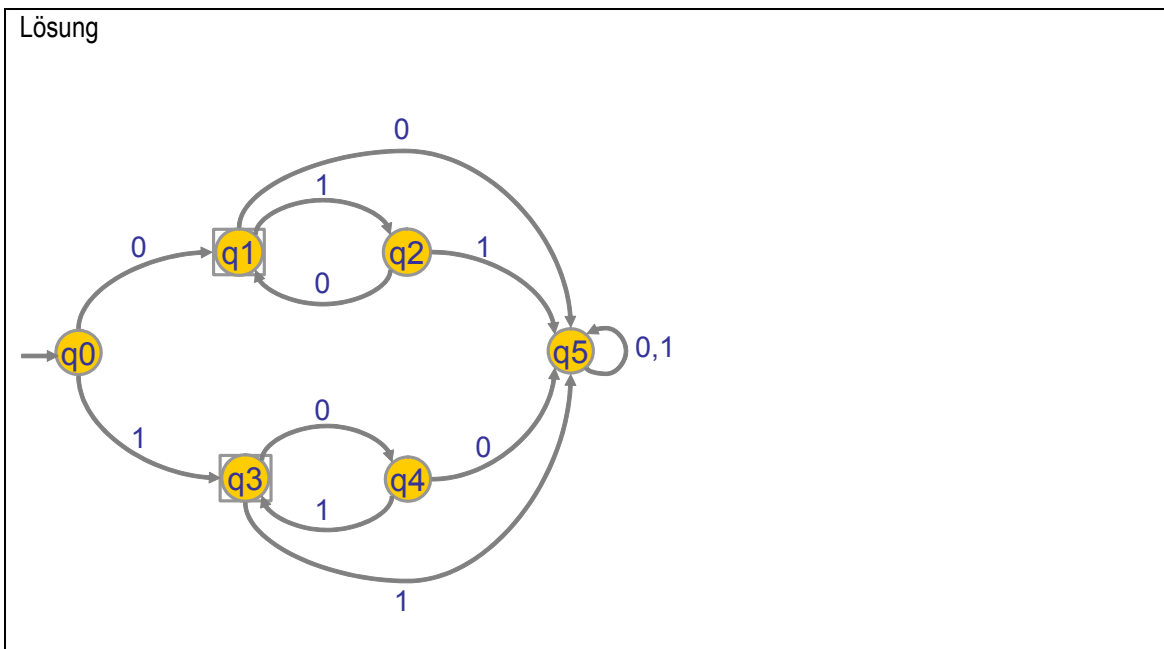
- Das Zeichen 0 tritt in jedem Wort der Sprache L entweder einmal mehr oder einmal weniger auf als das Zeichen 1.
- Die Zeichen 0 und 1 kommen in jedem Wort w immer abwechselnd vor (d.h., es gibt kein Wort $w \in L$, in dem die Zeichenfolge 00 oder 11 vorkommt).

Beispiel:

101 und 010 $\in L$

110 und 1010 $\notin L$

- a) Erstellen Sie einen vollständigen, deterministischen Akzeptor, der die Sprache L akzeptiert. **(5 Punkte)**



- b) Erstellen Sie eine Chomsky-Grammatik G vom Typ CH-3, die die Sprache L akzeptiert. Die Grammatik darf keine ϵ -Produktion (Produktion der Form $A \rightarrow \epsilon$) enthalten. **(5 Punkte)**

Lösung

$G = (\{0, 1\}, \{A, B, C, D, E\}, P, A)$

$P = \{$
 $A \rightarrow 0B \mid 1D \mid 0 \mid 1,$
 $B \rightarrow 1C,$
 $C \rightarrow 0B \mid 0,$
 $D \rightarrow 0E,$
 $E \rightarrow 1D \mid 1\}$

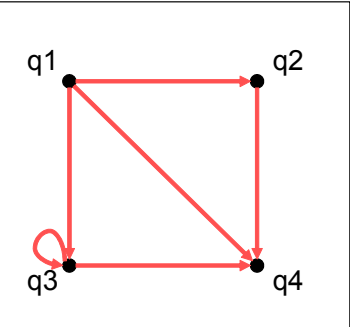
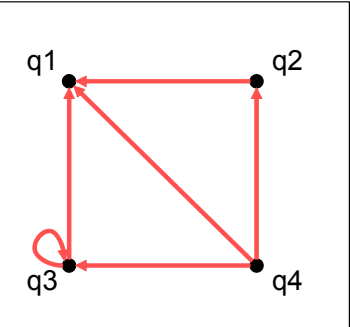
3 Graphen (8 Punkte)

a) Der gerichtete Graph $G_0 = (V_0, E_0)$ sei durch $V_0 = \{1, 2, 3, 4\}$ und die Adjazenzmatrix

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

gegeben.

Zeichnen Sie den Graphen G_0 und seinen dualen Graphen in die folgenden Kästen und geben Sie die Adjazenzmatrix für den dualen Graphen an: **(3 Punkte)**

Graph G_0	Dualer Graph zu G_0	Adjazenzmatrix zum dualen Graphen zu G_0
		$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

b) Bei dieser Teilaufgabe gibt es für jede korrekte Antwort 0,5 Punkte und für jede falsche Antwort werden 0,5 Punkte abgezogen. Nicht beantwortete Fragen werden mit 0 Punkten bewertet. Die Teilaufgabe wird mind. mit 0 Punkten bewertet.

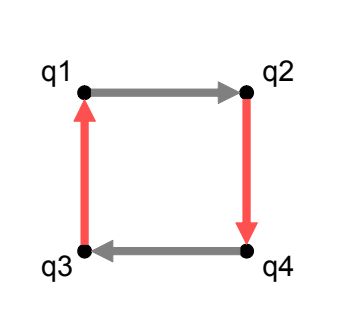
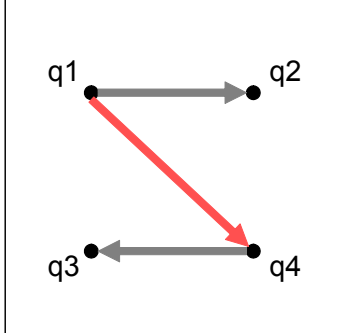
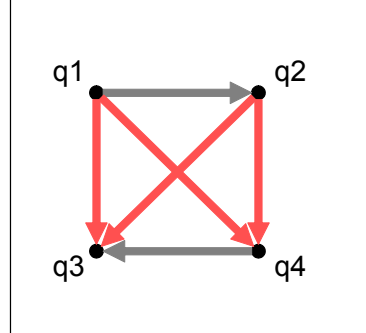
Enthält der Graph G_0 mindestens

(2 Punkte)

	Ja	nein
einen Zyklus	X	
einen Eulerschen Zyklus		X
einen Hamiltonschen Kreis		X
eine Ecke mit Ausgangsgrad 3	X	

c) Ergänzen Sie den folgenden Graphen, so dass

(3 Punkte)

		
---	---	--

er einen Hamiltonschen Kreis enthält.

er ein Baum ist.

jeder Knoten des Graphen einen anderen Eingangsgrad besitzt.

4 Java-Programm: Verständnis (8 Punkte)

a) Was berechnet die folgende Java-Methode?

(2 Punkte)

```
public int method1(int n) {
    if (n == 1) return 1;
    else return n + method1(n - 1);
}
```

Lösung

method1(n) berechnet die Summenfunktion $\sum_{i=1}^n i$

Gegeben seien die folgenden beiden Methoden:

```
public boolean method2(int n) {           // assertion:    n > 1
    for (int j = 2; j < n; j++) {
        if ((n % j) == 0) {
            return false;
        }
    }
    return true;
}
```

```
public void method3(int n) {              // assertion:    n > 1
    for (int i = 2; i < n; i++) {
        if method2(i) {
            Out.println(i);
        }
    }
}
```

b) Wann ist der in der Methode method2(n) verwendete Ausdruck $((n \% j) == 0)$ wahr?

(1 Punkt)

Lösung

Wenn n durch j ohne Rest teilbar ist.

c) Welche Eigenschaft des Parameters n wird durch die Methode method2(n) überprüft?

(3 Punkte)

Lösung

Es wird überprüft, ob n eine Primzahl ist.

d) Welche Ausgabe erfolgt beim Aufruf der Methode method3(n)?

(2 Punkte)

Lösung

method3(n) gibt alle Primzahlen aus, die kleiner als n und größer als 1 sind.

5 Java-Programm: Fehlersuche (7 Punkte)

Gegeben seien die Java-Methoden `pow1(x, y)` zur rekursiven und `pow2(x, y)` zur iterativen Berechnung von x^y sowie die Methode `fact(n)` zur Berechnung der Fakultät von n . Diese Methoden enthalten einige Fehler. Beschreiben Sie diese Fehler unter Verwendung von Java-Kommentaren in den folgenden Kästen. Die Kommentare dürfen hierbei in englischer oder deutscher Sprache verfasst werden.

Fehlerhafte Originalmethode `pow1()`:

(1 Punkt)

```
public static int pow1(int x, int y) {  
    if (y = 0) return 1;           // the boolean expression should be y == 0  
    else return x * pow1(x, y - 1); // one bracket is missing before x  
                                    // or one bracket needs to be deleted after y - 1  
}
```

Fehlerhafte Methode `pow2()`:

(3 Punkte)

```
public static int pow2(int x, y) {           // input variable y type should be int  
    long pow;                               // initial value for pow missing  
    for (int i == 0; i <= y; i++) {         // it should be i = 0  
        pow = pow * x                       // the middle expression should be i < y  
                                             // semicolon missing  
                                             // closing bracket } missing  
    return pow;                             // type cast (int) pow needed  
}
```

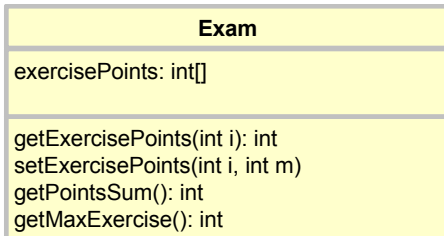
Fehlerhafte Methode `fact(n)`:

(3 Punkte)

```
public static int fact(int n){  
    if ((n == 1) && (n == 0)) {           // || should be used  
        return 1.0;                       // return value needs to be int or compatible  
                                             // bracket missing  
    Else                                  // else is written with a small letter at the beginning  
                                             // bracket missing  
        return n * fact(n - 1)           // semicolon missing  
    }  
}
```

6 Java-Programm: Programmentwicklung (11 Punkte)

Implementieren Sie die durch folgendes UML-Diagramm vorgegebene Klasse Exam entsprechend den aus der Vorlesung bekannten Programmierkonventionen. Die Kommentare dürfen dabei in englischer oder deutscher Sprache verfasst werden.



Objekte der Klasse Exam sollen die Punktezahlen für sechs Aufgaben aufnehmen können. Dazu besitzen sie als privates Attribut ein Array namens `exercisePoints`, in dem die Punkte als `int` gespeichert werden. Die Aufgaben werden im Rahmen dieser Aufgabe von 0 bis 5 durchnummeriert.

- Die Klasse besitzt einen parameterlosen Konstruktor, der das Array `exercisePoints` initialisiert und alle Einträge mit dem Wert 0 vorbelegt.
- Um den Zugriff auf die Werte des Arrays von außerhalb der Klasse zu ermöglichen, besitzt sie die Methoden `getExercisePoints(i)` und `setExercisePoints(i, m)`.
- Die beim Aufruf von `getExercisePoints(i)` übergebene Zahl `i` legt fest, von welcher Aufgabe die Punktezahl zurückgeliefert werden soll.
- Beim Aufruf von `setExercisePoints(i, m)` wird mit der übergebenen Zahl `i` die Aufgabe festgelegt, deren Wert mit der Zahl `m` belegt werden soll.
- Die Methode `getPointsSum()` liefert die Summe aller Punkte des Arrays `exercisePoints` zurück
- Die Methode `getMaxExercise()` liefert die Nummer der Aufgabe mit der höchsten Punktzahl des Arrays zurück.

Sie brauchen in dieser Aufgabe keine Ausnahmebehandlung für fehlerhafte Nutzung dieser Klasse realisieren und können davon ausgehen, dass nur gültige Zugriffe auf die Objekte der Klasse erfolgen.

Verwenden Sie zur Lösung dieser Aufgabe den Kasten auf dem folgenden Blatt.

Lösung zur Aufgabe 6 Java-Programm: Programmentwicklung

```
class Exam {
    private int[] exercisePoints;    // attribute containing the points of the exam

    Exam() {                          // class constructor
        exercisePoints = new int[6];
        for (int i = 0; i < 6; i++) {
            exercisePoints[i] = 0;
        }
    }

    public int getExercisePoints(int i) {    //get method for private attribute
        return exercisePoints[i];
    }

    public void setExercisePoints(int i, int m) {    //set method for private attribute
        exercisePoints[i] = m;
    }

    public int getPointsSum() {    // method to calculate the sum of points
        int sum = 0;
        for (int i = 0; i < 6; i++) {
            sum = sum + exercisePoints[i];
        }
        return sum;
    }

    public int getMaxExercise() { // method returns the exercise with the max. of points
        int max = exercisePoints[0];
        int number = 0;
        for (int i = 1; i < 6; i++) {
            if (exercisePoints[i] > max) {
                max = exercisePoints[i];
                number = i;
            }
        }
        return number;
    }
}
```

7 Wissensfragen (6 Punkte)

- a) Kreuzen Sie in folgender Tabelle an, ob die Aussagen wahr oder falsch sind. Richtige Antworten werden mit + 0,5 Punkten, falsche Antworten mit - 0,5 Punkten und nicht beantwortete Fragen mit 0 Punkten bewertet. Diese Teilaufgabe wird insgesamt mit mindestens 0 Punkten bewertet. **(3 Punkte)**

Aussage	Wahr	Falsch
Aus der Objektgleichheit folgt bei Java die Attributgleichheit.	X	
Handelt es sich bei Menge der ganzen Zahlen mit der Funktion x^y um einen Monoid?		X
Das Schlüsselwort <code>static</code> definiert bei Java Attributen, dass deren Wert konstant ist (Konstanten).		X
Eine abstrakte Klasse kann nicht instantiiert werden.	X	
Der Term $\mathbb{C} a \vee b$ der aus der Vorlesung bekannten booleschen Algebra ist äquivalent zum Ausdruck $a \Rightarrow b$.	X	
Das Schlüsselwort <code>Class</code> zur Beschreibung einer Klasse bei Java wird groß geschrieben.		X

- b) Nennen Sie drei aus der Vorlesung bekannte dynamische Datenstrukturen. **(1 Punkt)**

Liste
Baum
Graph

- c) Wie berechnet man den mittleren Entscheidungsgehalt H eines Zeichenvorrats? **(1 Punkt)**

$H = \sum p_i * \text{ld}(1/p_i)$

- d) Aus welchen vier Komponenten setzt sich laut der Vorlesung ein von-Neumann-Rechner zusammen? **(1 Punkt)**

Rechnerkern oder Prozessor oder CPU
Arbeitsspeicher
Peripherie
Bus