

Vorname

Nachname

Matrikel-Nr

Diese Klausur besteht aus 5 Aufgaben. Schreiben Sie jede Lösung möglichst auf die Vorderseite eines *neuen* Blattes (und lassen Sie die Rückseiten Ihrer Lösungsblätter *leer*).

**Aufgabe 1 (16 Punkte):** Geben Sie eine (kontextfreie) Grammatik an für die Menge aller Zahlen im 5-er-System, die (ohne Rest) durch 4 teilbar sind.

**Aufgabe 2 (16 Punkte):** Ein *Hex-Literal* ist ein nicht-leeres Wort über dem folgenden Alphabet von 22 Symbolen:

HexZiff = {0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f}

Ein Hex-Literal ist *nicht-ordentlich*, wenn es mindestens einen großen und mindestens einen kleinen Buchstaben enthält. Alle anderen Hex-Literale bezeichnen wir als *ordentlich*.

Beispiele für *ordentliche* Hex-Literale: 0, 123, 1A, 1a, ABC, abc, c9b8a7, 9C8B7A etc.

Beispiele für *nicht-ordentliche* Hex-Literale: aA, Bc, 12F34a, FF12345bb etc.

Geben Sie eine (kontextfreie) Grammatik für die Menge aller ordentlichen Hex-Literale an.

**Aufgabe 3 (16 Punkte):** In einem Gentle-Programm sei folgender Typ definiert:

```
1 'type' LISTE
2   leer
3   k(Zahl: INT, Rest: Liste)
```

Jeder Wert dieses Typs ist eine Liste, die 0 oder mehr INT-Werte enthält.

Definieren Sie ein Prädikat namens `anzPosNeg`, welches die Anzahl der positiven und die Anzahl der negativen Zahlen in einer Liste („auf einmal“) berechnet. Dabei soll die Zahl 0 als positiv gelten. Was für eine Art von Prädikat (action oder condition) sollte `anzPosNeg` sein? Wieviele Parameter sollte `anzPosNeg` haben und zu welchen Typen sollten die Parameter gehören?

**Aufgabe 4 (16 Punkte):** Definieren Sie zwei Prädikate, die den folgenden Spezifikationen entsprechen (der Typ `LISTE` wurde in der vorigen Aufgabe definiert):

```
4 'condition' kommtVor(Zahl: INT, Liste: LISTE)
5 -- Gelingt genau dann, wenn die Zahl in der Liste vorkommt
6
7 'condition' enthaelt(Liste1: LISTE, Liste2: LISTE)
8 -- Gelingt genau dann, wenn die Liste1 die Liste2 enthaelt,
9 -- d.h. wenn jede Zahl in der Liste2 auch in der Liste1 vorkommt
```

**Aufgabe 5 steht auf der Rückseite!**

**Aufgabe 5** (16 Punkte): Beantworten Sie die folgenden Fragen möglichst kurz, aber genau.

5.1. In Programmen, die in den Sprachen C, C++ oder Gentle geschrieben sind, unterscheidet man zwei Arten von Vereinbarungen: *Deklarationen* und *Definitionen*. Was ist dabei eine *Deklaration*? Geben Sie die in der Vorlesung behandelte Kurzdefinition (für den Begriff Deklaration) an.

5.2. Angenommen, Sie programmieren in Gentle einen Compiler und benötigen ein *Token*-Prädikat. Wo definieren Sie das Tokenprädikat? Welche Mittel (Sprachen) benutzen Sie für die Definition?

5.3. Was befiehlt der Programmierer dem Ausführer mit einem *Ausdruck* (z.B. mit dem Ausdruck  $x + 1$ )?

5.4. Im Laufe des Semesters haben Sie einen Compiler entwickelt, der Programme einer Sprache namens Alg in Assemblerbefehle der JVM (Java Virtual Machine) übersetzt. In was übersetzt Ihr Compiler einen *Ausdruck* der Sprache Alg? Was leistet die Übersetzung eines Ausdrucks?

5.5. In einigen Programmiersprachen (darunter Gentle und Prolog) spielen *Terme* eine wichtige Rolle. Einen Term kann man auf zwei ganz verschiedene („gegensätzliche“) Weisen benutzen und bezeichnet ihn dann mit verschiedenen Fachbegriffen (einen Term kann man benutzen als ... oder als ...). Geben Sie die beiden Fachbegriffe an.

5.6. Beschreiben Sie (möglichst kurz) die beiden Operationen, die man auf Terme anwenden kann. Wie heissen die Operationen? Was braucht man, um sie auszuführen? Was kommt als Ergebnis heraus? Besonderheiten?

**Lösung 1 (16 Punkte):** Eine (kontextfreie) Grammatik für die Menge aller Zahlen im 5-er-System, die (ohne Rest) durch 4 teilbar sind.

Das Startsymbol der folgenden Grammatik ist RK0:

```

R01: RK0 -> 0      R11: RK1 -> RK1 0      R21: RK3 -> RK3 0
R02: RK1 -> 1      R12: RK2 -> RK1 1      R22: RK0 -> RK3 1
R03: RK2 -> 2      R13: RK3 -> RK1 2      R23: RK1 -> RK3 2
R04: RK3 -> 3      R14: RK0 -> RK1 3      R24: RK2 -> RK3 3
R05: RK0 -> 4      R15: RK1 -> RK1 4      R25: RK3 -> RK3 4

R06: RK0 -> RK0 0   R16: RK2 -> RK2 0
R07: RK1 -> RK0 1   R17: RK3 -> RK2 1
R08: RK2 -> RK0 2   R18: RK0 -> RK2 2
R09: RK3 -> RK0 3   R19: RK1 -> RK2 3
R10: RK0 -> RK0 4   R20: RK2 -> RK2 4

```

**Lösung 2 (16 Punkte):** Eine Grammatik für die Menge aller ordentlichen Hex-Literale:

```

KB -> 'a'      GB -> 'A'      ZIFF -> '0'
KB -> 'b'      GB -> 'B'      ZIFF -> '1'
...
KB -> 'f'      GB -> 'F'      ZIFF -> '9'

ZOK -> Ziff      -- ZOK: Ziffer oder Kleinbuchstabe
ZOK -> KB

ZOG -> ZIFF      -- ZOG: Ziffer oder Großbuchstaben
ZOG -> GB

ZOK_FO -> ZOK      -- ZOK_FO: Folge von ZOKs
ZOK_FO -> ZOK ZOK_FO

ZOG_FO -> ZOG      -- ZOG_FO: Folge von ZOGs
ZOG_FO -> ZOG ZOG_FO

OHL -> ZOG_FO      -- OHL: Ordentliches Hex-Literal, Startsymbol!
OHL -> ZOK_FO

```

**Lösung 3 (16 Punkte):** Definieren Sie ein Prädikat namens `anzPosNeg`, welches die Anzahl der positiven und die Anzahl der negativen Zahlen in einer Liste („auf einmal“) berechnet. Dabei soll die Zahl 0 als positiv gelten.

```

10 'action' anzPosNeg(Liste: LISTE -> AnzPos: INT, AnzNeg: INT)
11   'rule' anzPosNeg(leer -> 0, 0):
12   'rule' anzPosNeg(k(N, R) -> P+1, N):
13     ge(N, 0)
14     anzPosNeg(R -> P, N)
15   'rule' anzPosNeg(k(N, R) -> P, N+1):
16     anzPosNeg(R -> P, N)

```

**Lösung 4 (16 Punkte):** Definieren Sie zwei Prädikate, die den folgenden Spezifikationen entsprechen:

```

17 'condition' kommtVor(Zahl: INT, Liste: LISTE)
18 -- Gelingt genau dann, wenn die Zahl in der Liste vorkommt
19 'rule' kommtVor(N1, k(N2, R)):
20   eq(N1, N2)
21 'rule' kommtVor(N1, k(N2, R)):
22   kommtVor(N1, R)
23
24 'condition' enthaelt(Liste1: LISTE, Liste2: LISTE)
25 -- Gelingt genau dann, wenn die Liste1 die Liste2 enthaelt,
26 -- d.h. wenn jede Zahl in der Liste2 auch in der Liste1 vorkommt
27 'rule' enthaelt(L, leer): .
28 'rule' enthaelt(L, k(N, R)):
29   kommtVor(N, L)
30   enthaelt(L, R)

```

**Lösung 4 (16 Punkte):** Ein Bedingungs-Prädikat (condition predicate) namens `equals` mit zwei Eingabeparametern vom Typ `TBAUM`. Ein Aufruf `equal(B1, B2)` glückt genau dann, wenn die beiden Bäume `B1` und `B2` gleich sind.

```
1 'condition' equals(B1: TBAUM, B2: TBAUM)
2   'rule' equals(leer, leer): .
3   'rule' equals(k(Z1, B11, B12, B13), k(Z2, B21, B22, B23)):
4       eq(Z1, Z2)
5       equals(B11, B21)
6       equals(B12, B22)
7       equals(B13, B23)
```

**Lösung 5 (16 Punkte):** Beantworten Sie die folgenden Fragen möglichst kurz, aber genau.

5.1. In Programmen, die in den Sprachen C, C++ oder Gentle geschrieben sind, unterscheidet man zwei Arten von Vereinbarungen: **Deklarationen** und **Definitionen**. Was ist dabei eine **Deklaration**? Geben Sie möglichst nur die in der Vorlesung behandelte Kurzdefinition (für den Begriff Deklaration) an.

Eine Deklaration ist ein Versprechen (des Programmierers an den Ausführer), die deklarierte Größe (irgendwo im Programm) zu definieren.

5.2. Angenommen, Sie programmieren in Gentle einen Compiler und benötigen ein **Token-Prädikat**. Wo definieren Sie das Tokenprädikat? Welche Sprachen benutzen Sie für die Definition?

Ein Token-Prädikat definiert man in einer `.t`-Datei. Dabei verwendet man einen regulären Ausdruck und C-Befehle.

5.3. Was befiehlt der Programmierer dem Ausführer mit einem **Ausdruck** (z.B. mit dem Ausdruck `x + 1`)?

Mit einem Ausdruck befiehlt der Programmierer dem Ausführer, einen Wert zu berechnen.

5.4. Im Laufe des Semesters haben Sie einen Compiler entwickelt, der Programme einer Sprache namens `Alg` in Assemblerbefehle der JVM (Java Virtual Machine) übersetzt. In was übersetzt Ihr Compiler einen **Ausdruck** der Sprache `Alg`? Was leistet die Übersetzung eines Ausdrucks?

Jeder Ausdruck wird in eine Folge von Assemblerbefehlen übersetzt, die bewirken, dass der Wert des Ausdrucks auf den Stapel (der JVM) gelegt wird.

5.5. In einigen Programmiersprachen (darunter Gentle und Prolog) spielen Terme eine wichtige Rolle. Einen Term kann man auf zwei ganz verschiedene („gegensätzliche“) Weisen benutzen und bezeichnet ihn dann mit verschiedenen Fachbegriffen. Einen Term kann man benutzen als ... oder als ... . Geben Sie die beiden Fachbegriffe an.

Man benutzt Terme als **Ausdrücke** und als **Muster**.

5.6. Beschreiben Sie (ganz kurz) die beiden Operationen, die man auf Terme anwenden kann. Wie heissen die Operationen? Was braucht man, um sie auszuführen? Was kommt als Ergebnis heraus? Besonderheiten?

Operation Auswertung: Ein Ausdruck und eine Variablenbelegung, Ergebnis ist ein Wert.

Operation Musterabgleich: Ein Muster und ein Wert, Ergebnis ist eine Variablenbelegung.

Besonderheiten: Ein Musterabgleich kann gelingen oder misslingen.