

Vorname

Nachname

Matrikel-Nr

Diese Klausur besteht aus **4 Aufgaben**. Schreiben Sie jede Lösung möglichst auf die Vorderseite eines *neuen* Blattes (und lassen Sie die Rückseiten Ihrer Lösungsblätter *leer*).

Aufgabe 1 (20 Punkte): Geben Sie eine (kontextfreie) Grammatik an für die Menge aller Zahlen im 3-er-System, die (ohne Rest) durch 5 teilbar sind.

Aufgabe 2 (20 Punkte): Sei `PARAM` ein Zwischensymbol, aus dem man Parameter ableiten kann. Gehen Sie davon aus, dass die dazu nötigen Regeln vorgegeben und fertig sind (auch wenn sie hier nicht ausdrücklich angegeben werden).

Eine *Parameterliste* besteht aus Parametern, die durch Kommas voneinander getrennt und in runde Klammern eingefaßt sind, z.B. `()` oder `(p1)` oder `(p1, p2, p3)` etc.

2.1. Geben Sie eine Grammatik für die Sprache aller Parameterlisten (einschließlich der leeren Parameterliste `()`) an. Startsymbol: `PLML0` ("Parameterliste, Mindestlänge 0")

2.2. Geben Sie eine Grammatik an für die Sprache aller nicht-leeren Parameterlisten an. Startsymbol: `PLML1` ("Parameterliste, Mindestlänge 1").

Aufgabe 3 (20 Punkte): In einem Gentle-Programm seien folgende Baum-Typen definiert:

```

1  'type' ROT
2      rLeer
3      rBR(BLAU, ROT)
4      rGRR(GRUEN, ROT, ROT)
5
6  'type' BLAU
7      bLeer
8      bGR(GRUEN, ROT)
9
10 'type' GRUEN
11     gLeer
12     gRG(ROT, GRUEN)

```

Beachten Sie: Ein Baum des Typs `ROT` kann Unterbäume der Typen `ROT`, `BLAU` und `GRUEN` besitzen. Für die anderen beiden Baum-Typen gilt entsprechendes. Definieren Sie die folgenden drei Prädikate:

```

13 'action' anzInRot (ROT -> AnzRote: INT, AnzBlaue: INT, AnzGruene: INT)
14 'action' anzInBlau (BLAU -> AnzRote: INT, AnzBlaue: INT, AnzGruene: INT)
15 'action' anzInGruen (GRUEN -> AnzRote: INT, AnzBlaue: INT, AnzGruene: INT)

```

Das Prädikat `anzInRot` hat einen Baum des Typs `ROT` als Eingabeparameter und liefert die Anzahl der roten, der blauen und der grünen Unterbäume dieses Baums. Beachten Sie dabei, dass jeder Baum auch als Unterbaum von sich selbst gilt (so wie jede Menge als Untermenge von sich selbst gilt). Beispiele:

```

16 anzInRot(rLeer -> R, B, G)
17     belegt die Variablen R, B, G mit den Werten 1, 0, 0
18 anzInRot(rBR(bLeer, rLeer) -> R, B, G)
19     belegt die Variablen R, B, G mit den Werten 2, 1, 0
20 anzInRot(rGRR(gLeer, rBR(bLeer, rLeer), rLeer))
21     belegt die Variablen R, B, G mit den Werten 4, 1, 1

```

Für die anderen beiden Prädikate (`anzInBlau` und `anzInGruen`) gilt Entsprechendes.

Aufgabe 4 (20 Punkte): Beantworten Sie die folgenden Fragen möglichst kurz, aber genau.

4.1. Beschreiben Sie kurz den Unterschied zwischen den beiden Parsergeneratoren `yacc` und `acent`.

4.2. Geben Sie eine formale Sprache an, die man mit keiner Typ-2-Grammatik (kontextfreien Grammatik), wohl aber mit einer Typ-1-Grammatik (kontextsensitiven Grammatik) beschreiben kann.

4.3. Geben Sie eine formale Sprache an, die man mit keiner Grammatik (auch nicht mit einer Typ-0-Grammatik) beschreiben kann.

4.4. Die kontextfreie Grammatik der Sprache Java beschreibt nicht genau die Sprache Java sondern eine etwas andere Sprache Java' ("Java Strich"). Welche Worte gehören zu beiden Sprachen (Java und Java')? Welche Worte gehören nur zu einer der beiden Sprachen (Java bzw. Java')?

4.5. 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.

4.6. 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 (20 Punkte): Eine (kontextfreie) Grammatik für die Menge aller Zahlen im 3-er-System, die (ohne Rest) durch 5 teilbar sind. Das Startsymbol der folgenden Grammatik ist RK0:

```

R01: RK0 -> 0
R02: RK1 -> 1
R03: RK2 -> 2

R04: RK0 -> RK0 0
R05: RK1 -> RK0 1
R06: RK2 -> RK0 2

R07: RK3 -> RK1 0
R08: RK4 -> RK1 1
R09: RK0 -> RK1 2

R10: RK1 -> RK2 0
R11: RK2 -> RK2 1
R12: RK3 -> RK2 2

R13: RK4 -> RK3 0
R14: RK0 -> RK3 1
R15: RK1 -> RK3 2

R16: RK2 -> RK4 0
R17: RK3 -> RK4 1
R18: RK4 -> RK4 2

```

Lösung 2 (20 Punkte):

2.1. Eine Grammatik für die Menge aller Parameterlisten (einschließlich der leeren):

```

R01: PLML0 -> "(" LISTE0 ")"
R02: LISTE0 ->
R03: LISTE0 -> LISTE1
R04: LISTE1 -> PARAM
R05: LISTE1 -> PARAM "," LISTE1

```

2.2. Eine Grammatik für die Menge aller nicht-leeren Parameterlisten:

```

R06: PLML1 -> "(" LISTE1 ")"

```

Lösung 3 (20 Punkte): In einem Gentle-Programm seien folgende Baum-Typen definiert:

```

22 'type' ROT
23   rLeer
24   rBR(BLAU, ROT)
25   rGRR(GRUEN, ROT, ROT)
26
27 'type' BLAU
28   bLeer
29   bGR(GRUEN, ROT)
30
31 'type' GRUEN
32   gLeer
33   gRG(ROT, GRUEN)

```

Die Definitionen der drei Prädikate:

```

34 'action' anzInRot (ROT -> AnzRote: INT, AnzBlaue: INT, AnzGruene: INT)
35   -- Zaehlt die Anzahl der roten, der blauen und der gruenen Unterbaeume
36   -- in einem roten Baum:
37   'rule' anzInRot(rLeer -> 1, 0, 0):
38   'rule' anzInRot(rBR(B, R) -> 1+BR+RR, BB+RB, BG+RG):
39     anzInBlau (B -> BR, BB, BG)
40     anzInRot (R -> RR, RB, RG)
41   'rule' anzInRot(rGRR(G, R1, R2) -> GR+R1R+R2R+1, GB+R1B+R2B, GG+R1G+R2G):
42     anzInGruen(G -> GR, GB, GG)
43     anzInRot (R1 -> R1R, R1B, R1G)
44     anzInRot (R2 -> R2R, R2B, R2G)
45

```

```

46 'action' anzInBlau (BLAU -> AnzRote: INT, AnzBlaue: INT, AnzGruene: INT)
47 -- Zaehlt die Anzahl der roten, der blauen und der gruenen Unterbaeume
48 -- in einem blauen Baum:
49 'rule' anzInBlau(bLeer -> 0, 1, 0):
50 'rule' anzInBlau(bGR(G, R) -> GR+RR, 1+GB+RB, GG+RG):
51     anzInGruen(G -> GR, GB, GG)
52     anzInRot (R -> RR, RB, RG)
53
54 'action' anzInGruen(GRUEN -> AnzRote: INT, AnzBlaue: INT, AnzGruene: INT)
55 -- Zaehlt die Anzahl der roten, der blauen und der gruenen Unterbaeume
56 -- in einem gruenen Baum:
57 'rule' anzInGruen(gLeer -> 0, 0, 1):
58 'rule' anzInGruen(gRG(R, G) -> RR+GR, RB+GB, 1+RG+GG):
59     anzInRot (R -> RR, RB, RG)
60     anzInGruen(G -> GR, GB, GG)

```

Lösung 4 (20 Punkte): Beantworten Sie die folgenden Fragen möglichst kurz, aber genau.

4.1. Beschreiben Sie kurz den Unterschied zwischen den beiden Parsergeneratoren `yacc` und `acc-cent`.

Der `yacc` kann nur für bestimmte (kontextfreie) Grammatiken ("LR-Grammatiken") einen Parser generieren. Der `acc-cent` kann für jede (kontextfreie) Grammatik einen Parser generieren.

4.2. Geben Sie eine formale Sprache an, die man mit keiner Typ-2-Grammatik (kontextfreien Grammatik), wohl aber mit einer Typ-1-Grammatik (kontextsensitiven Grammatik) beschreiben kann.

Z.B. die Sprache $L_3 = \{a^n b^n c^n \mid n \geq 1\}$

4.3. Geben Sie eine formale Sprache an, die man mit keiner Grammatik (auch nicht mit einer Typ-0-Grammatik) beschreiben kann.

Die Sprache aller nicht-haltenden Java-Programme.

4.4. Die kontextfreie Grammatik der Sprache Java beschreibt nicht genau die Sprache Java sondern eine etwas andere Sprache Java' ("Java Strich"). Welche Worte gehören zu beiden Sprachen (Java und Java')? Welche Worte gehören nur zu einer der beiden Sprachen (Java und Java')?

Zu beiden Sprachen gehören alle (syntaktisch) korrekten Java-Programme. Zur Sprache Java' gehören zusätzlich Symbolfolgen, die einem Java-Programm sehr ähnlich sehen, aber bestimmte Kontextbedingungen nicht einhalten.

4.5. 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.

Einen Term kann man als Ausdruck oder als Muster benutzen.

4.6. 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?

Auf Terme kann man die Operationen Auswertung und Musterabgleich anwenden:

Auswertung : Ausdruck x Var.-Beleg. ergibt einen Wert.

Musterabgleich: Muster x Wert ergibt eine Var.-Beleg.

Eine Auswertung gelingt immer. Ein Musterabgleich kann gelingen oder misslingen.