

Vorname

Nachname

Matrikel-Nr

Schreiben Sie jede Lösung möglichst auf die Vorderseite eines neuen Blattes (und lassen Sie die Rückseiten Ihrer Lösungsblätter leer).

**Aufgabe 1** (10 Punkte): Geben Sie eine kontextfreie Grammatik (d.h. eine Typ2-Grammatik) an für die Menge aller Zahlen im 5-er-System, die den Rest 2 ergeben wenn man sie durch 3 teilt.

**Aufgabe 2** (20 Punkte): In einem Gentle-Programm sei folgender Typ definiert:

```
1 'type' BAUM
2   leer
3   k(Zahl1: INT, Zahl2: INT, LinkerBaum: BAUM, RechterBaum: BAUM)
```

Die Werte dieses Typs sind binäre Bäume, die aus null oder mehr Knoten bestehen. Jeder Knoten enthält als "Nutzdaten" zwei Zahlen (vom Typ INT).

Vereinbaren Sie ein Aktions-Prädikat namens **anzGleiche**, welches die Anzahl der Knoten mit zwei gleichen Zahlen in einem Baum ermittelt. Hier die Spezifikation des Prädikates:

```
4 'action' anzGleiche(BAUM -> INT)
```

Vereinbaren Sie geeignete Regeln für dieses Prädikat.

**Aufgabe 3** (30 Punkte): In einem Gentle-Programm seien folgende Typen definiert:

```
5 'type' ROT
6   rLeer
7   rBR(BLAU, ROT)
8   rGRR(GRUEN, ROT, ROT)
9
10 'type' BLAU
11   bLeer
12   bGR(GRUEN, ROT)
13
14 'type' GRUEN
15   gLeer
16   gRG(ROT, GRUEN)
```

Vereinbaren Sie ein Aktions-Prädikat namens **anzRotInRot**, welches die folgende Spezifikation realisiert:

```
17 'action' anzRotInRot(ROT -> INT)
18   -- Zaehlt die Anzahl der roten Unterbaeume in einem roten Baum.
19   -- Hinweis 1: Jeder Baum zaehlt als ("uneigentlicher") Unterbaum von sich selbst.
20   -- Hinweis 2: Auch leere Baeume sind "richtige Baeume"!
```

Diese Aufgabe ist besonders leicht zu lösen, wenn man nicht nur das Prädikat **anzRotInRot** vereinbart, sondern noch ein paar geeignete "Hilfsprädikate". Es ist ausdrücklich erlaubt, solche zusätzlichen Prädikate zu vereinbaren und aufzurufen.

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

- 4.1. Welche Prädikate eines Gentle-Programms werden einem Parsergenerator (yacc bzw. bison) übergeben, damit der daraus einen Parser generiert?
- 4.2. Welche Prädikate eines Gentle-Programms werden einem Lexergenerator (lex bzw. flex) übergeben, damit der daraus einen Lexer generiert?
- 4.3. Welche weiteren "kleinen Bestandteile eines Gentle-Programms" werden dem Lexergenerator übergeben, damit der daraus einen Lexer generiert?
- 4.4. Was ist der wesentliche Unterschied zwischen einem Aktionsprädikat (action predicate) und einem Bedingungsprädikat (condition predicate)?
- 4.5. Wie definiert man ein Tokenprädikat? Welche Sprachen benutzt man dazu? Beschreiben Sie nur kurz die grundsätzliche Vorgehensweise, keine "fitzeligen Einzelheiten".
- 4.6. Die Sprachen Prolog und Gentle ähneln sich stark, unterscheiden sich aber auch an einigen wichtigen Stellen. Welcher Unterschied bewirkt vor allem, dass Gentle-Programme viel schneller sind als Prolog-Programme?

**Lösung 1** (10 Punkte): Eine kontextfreie Grammatik (d.h. eine Typ2-Grammatik) für die Menge aller Zahlen im 5-er-System, die den Rest 2 ergeben wenn man sie durch 3 teilt.

Das Startsymbol der folgenden Grammatik ist **RK2**:

```

1 RK0 -> 0
2 RK1 -> 1
3 RK2 -> 2
4 RK0 -> 3
5 RK1 -> 4
6
7 RK0 -> RK0 0
8 RK1 -> RK0 1
9 RK2 -> RK0 2
10 RK0 -> RK0 3
11 RK1 -> RK0 4
12
13 RK2 -> RK1 0
14 RK0 -> RK1 1
15 RK1 -> RK1 2
16 RK2 -> RK1 3
17 RK0 -> RK1 4
18
19 RK1 -> RK2 0
20 RK2 -> RK2 1
21 RK0 -> RK2 2
22 RK1 -> RK2 3
23 RK2 -> RK2 4

```

**Lösung 2** (20 Punkte): Ein Aktions-Prädikat namens **anzGleiche**, welches die Anzahl der Knoten mit zwei gleichen Zahlen in einem Baum ermittelt.

```

24 'action' anzGleiche(BAUM -> INT)
25   'rule' anzGleiche(leer -> 0): .
26   'rule' anzGleiche(k(Z1, Z2, B1, B2) -> 1+N1+N2):
27     eq(Z1, Z2)
28     anzGleiche(B1 -> N1)
29     anzGleiche(B2 -> N2)
30   'rule' anzGleiche(k(Z1, Z2, B1, B2) -> N1+N2):
31     anzGleiche(B1 -> N1)
32     anzGleiche(B2 -> N2)

```

**Lösung 3** (30 Punkte): Ein Aktions-Prädikat namens **anzRotInRot**: und zwei Hilfsprädikate namens **anzRotInBlau** und **anzRotInGruen**:

```

33 'action' anzRotInRot (ROT -> INT)
34   -- Zaehlt die Anzahl der roten Unterbaeume in einem
35   -- roten Baum:
36   'rule' anzRotInRot(rLeer -> 1): .
37
38   'rule' anzRotInRot(rBR(B, R) -> A1+A2+1):
39     anzRotInBlau(B -> A1)
40     anzRotInRot (R -> A2).
41
42   'rule' anzRotInRot(rGRR(G, R1, R2) -> A1+A2+A3+1):
43     anzRotInGruen(G -> A1)
44     anzRotInRot (R1 -> A2)
45     anzRotInRot (R1 -> A3).
46
47 'action' anzRotInBlau (BLAU -> INT)
48   -- Zaehlt die Anzahl der roten Unterbaeume in einem
49   -- blauen Baum:
50   'rule' anzRotInBlau(bLeer -> 0): .
51
52   'rule' anzRotInBlau(bGR(G, R) -> A1+A2):
53     anzRotInGruen(G -> A1)
54     anzRotInRot (R -> A2)

```

```
55
56 'action' anzRotInGruen(GRUEN -> INT)
57   -- Zaehlt die Anzahl der roten Unterbaeume in einem
58   -- gruenen Baum:
59   'rule' anzRotInGruen(gLeer -> 0): .
60
61   'rule' anzRotInGruen(gRG(R, G) -> A1+A2):
62     anzRotInRot (R -> A1)
63     anzRotInGruen(G -> A2)
```

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

4.1. Welche Prädikate eines Gentle-Programms werden einem Parsergenerator (yacc bzw. bison) übergeben, damit der daraus einen Parser generiert?

**Die Zwischensymbolprädikate (nonterm predicates).**

4.2. Welche Prädikate eines Gentle-Programms werden einem Lexergenerator (lex bzw. flex) übergeben, damit der daraus einen Lexer generiert?

**Die Endsymbolprädikate (token predicates).**

4.3. Welche weiteren "kleinen Bestandteile eines Gentle-Programms" werden dem Lexergenerator übergeben, damit der daraus einen Lexer generiert?

**Alle String-Literale, die in den Regeln von Zwischensymbolprädikaten vorkommen, und alle .t-Dateien und alle .b-Dateien.**

4.4. Was ist der wesentliche Unterschied zwischen einem Aktionsprädikat (action predicate) und einem Bedingungsprädikat (condition predicate)?

**Wenn ein Aufruf eines Aktionsprädikats mißlingt, dann ist das ein schwerwiegender Programmierfehler und bewirkt, dass die Ausführung des betreffenden Gentle-Programms (des Compilers) sofort abgebrochen wird. Wenn ein Aufruf eines Bedingungsprädikats mißlingt, dann ist das "ganz normal" und führt nicht zum Abbruch der Compilerausführung.**

4.5. Wie definiert man ein Endsymbolprädikat (token predicate)? Welche Sprachen benützt man dazu? Beschreiben Sie nur kurz die grundsätzliche Vorgehensweise, keine "fitzeligen Einzelheiten".

**Ein Endsymbolprädikat (token predicate) definiert man, indem man eine .t-Datei erstellt. In diese Datei schreibt man einen regulären Ausdruck und C-Befehle.**

4.6. Die Sprachen Prolog und Gentle ähneln sich stark, unterscheiden sich aber auch an einigen wichtigen Stellen. Welcher Unterschied bewirkt vor allem, dass Gentle-Programme viel schneller sind als Prolog-Programme?

**Ein Prolog-Ausführer arbeitet mit tiefem Wiederaufsetzen (deep backtracking), eine Gentle-Ausführer arbeitet dagegen mit flachem Wiederaufsetzen (shallow backtracking).**