

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 4-er-System, die den Rest 1 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 **anzUngleiche**, welches die Anzahl der Knoten mit zwei ungleichen Zahlen in einem Baum ermittelt. Hier die Spezifikation des Prädikates:

```
4 'action' anzUngleiche(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 **anzBlauInGruen**, welches die folgende Spezifikation realisiert:

```
17 'action' anzBlauInGruen(GRUEN -> INT)
18   -- Zaehlt die Anzahl der blauen Unterbaeume in einem gruenen Baum.
19   -- Jeder gruene Baum G enthaelt mindestens einen gruenen Unterbaum (naemlich
20   -- sich selbst). Falls der Baum G nicht leer ist, enthaelt er aussser sich
21   -- selbst eventuell noch weitere gruenen Unterbaeume.
```

Diese Aufgabe ist besonders leicht zu lösen, wenn man nicht nur das Prädikat **anzBlauInGruen** 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 4-er-System, die den Rest 1 ergeben wenn man sie durch 3 teilt.

Das Startsymbol der folgenden Grammatik ist RK1:

```

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

```

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

```

20 'action' anzUngleiche(BAUM -> INT)
21   'rule' anzUngleiche(leer -> 0): .
22   'rule' anzUngleiche(k(Z1, Z2, B1, B2) -> 1+N1+N2):
23     ne(Z1, Z2)
24     anzUngleiche(B1 -> N1)
25     anzUngleiche(B2 -> N2)
26   'rule' anzUngleiche(k(Z1, Z2, B1, B2) -> N1+N2):
27     anzUngleiche(B1 -> N1)
28     anzUngleiche(B2 -> N2)

```

**Lösung 3** (30 Punkte): Ein Aktions-Prädikat namens **anzBlauInGruen**: und zwei Hilfsprädikate namens **anzBlauInRot** und **anzBlauInBlau**:

```

29 'action' anzBlauInGruen (GRUEN -> INT)
30   -- Zaehlt die Anzahl der blauen Unterbaeume in einem
31   -- gruenen Baum:
32   'rule' anzBlauInGruen(gLeer -> 0): .
33
34   'rule' anzBlauInGruen(gRG(R, G) -> A1+A2):
35     anzRotInRot (R -> A1)
36     anzRotInGruen (G -> A2).
37
38 'action' anzBlauInRot (ROT -> INT)
39   -- Zaehlt die Anzahl der blauen Unterbaeume in einem
40   -- roten Baum:
41   'rule' anzBlauInRot(rLeer -> 0): .
42
43   'rule' anzBlauInRot(rBR(B, R) -> A1+A2):
44     anzBlauInBlau(B -> A1)
45     anzBlauInRot (R -> A2).
46
47   'rule' anzBlauInRot(rGRR(G, R1, R2) -> A1+A2+A3):
48     anzBlauInGruen(G -> A1)
49     anzBlauInRot (R1 -> A2)
50     anzBlauInRot (R1 -> A3).
51

```

```
52 'action' anzBlauInBlau(BLAU -> INT)
53   -- Zaehlt die Anzahl der blauen Unterbaeume in einem
54   -- blauen Baum:
55   'rule' anzBlauInBlau(bLeer -> 1): .
56
57   'rule' anzBlauInBlau(bGR(G, R) -> A1+A2+1):
58     anzBlauInGruen(G -> A1)
59     anzBlauInRot  (R -> 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).**