Vorname Nachname Matrikel-Nr.

Diese Klausur besteht aus **4 Aufgaben**. Schreiben Sie jede Ihrer Lösungen auf die Vorderseite eines *neuen* Blattes (und lassen Sie die Rückseiten Ihrer Lösungsblätter *leer*). Die Lösung für Aufgabe 2 können Sie auch direkt hinter die Fragen auf dieses Klausurblatt schreiben.

Aufgabe 1 (20 Punkte): Geben Sie eine (kontextfreie, Typ-2-) Grammatik an für die Menge aller natürlichen Zahlen (0, 1, 2, ...) im **3-er-System**, die (ohne Rest) **durch 5 teilbar** sind. Startsymbol `RK0` (wie "Restklasse Null").

Aufgabe 2 (20 Punkte): Betrachten Sie die folgende Grammatik für Ausdrücke (Endsymbole sind in doppelte Anführungszeichen eingefaßt, z.B. `"#1"` oder `"c"`. `AUS1` ist das Startsymbol):

```
AUS1 : AUS2 "#1" AUS1
AUS1 : AUS2

AUS2 : AUS2 "#2" AUS3
AUS2 : AUS3

AUS3 : "#3" AUS3
AUS3 : AUS4

AUS4 : "#4" AUS5
AUS4 : AUS5

AUS5 : LIT
AUS5 : "(" AUS1 ")"

LIT  : "a"
LIT  : "b"
LIT  : "c"
```

In dieser Grammatik kommen die Operatoren `#1`, `#2`, `#3`, `#4` vor. Geben Sie als Antworten auf die folgenden Fragen die beffenden Operatoren an:

2.01. Alle zweistelligen Operatoren?

2.02. Alle einstelligen Operatoren?

2.03. Alle linksassoziativen Operatoren?

2.04. Alle rechtsassoziativen Operatoren?

2.05. Welcher Operator bindet am stärksten?

2.06. Welcher Operator bindet am schwächsten?

Welche der folgenden Worte kan man aus obiger Grammatik ableiten (JA) und welche nicht (NEIN)?

2.07. `#3 #3 a`

2.08. `#4 #4 b`

2.09. `a #2 b #1 c`

2.10. `c #1 c #2 a`

Aufgabe 3 (40 Punkte):

3.1. Schreiben Sie ein Prädikat entsprechend der folgenden Spezifikation:

```
proc listeDerBesonderen(L:int[] -> BL:int[])
   // Die Liste BL enthaelt alle besonderen Elemente der Liste L.
   // Welche Elemente "besonders" sind, entscheidet das Praedikat
   //
   // condition istBesonders(N:int)
   //    // Gelingt, wenn die Zahl N besonders ist
   //
   // Nehmen Sie an, dass das Praedikat istBesonders vorgegeben ist, so dass
   // Sie es aufrufe können.
```

3.2. Schreiben Sie ein Prädikat entsprechend der folgenden Spezifikation:

```
condition keineZwillinge(L:int[])
   // Gelingt genau dann wenn L keine Zwillingspaare enthaelt. Ein Zwillingspaar
   // besteht aus zwei gleichen, benchbarten Elementen.
   // Beispiele:
   // Die Liste int[1, 2, 1, 2] enthaelt kein Zwillingspaar
   // Die Liste int[1, 2, 2, 1] enthaelt ein Zwillingspaar
```

3.3. Schreiben Sie ein Prädikat entsprechend der folgenden Spezifikation:

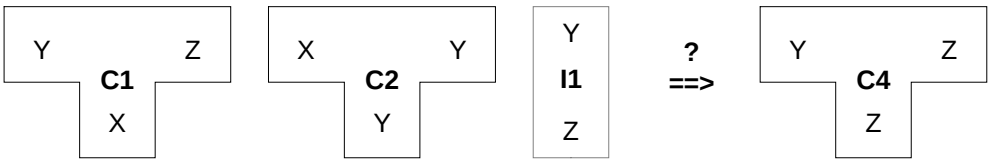
```
proc aus2mach1(L1:int[], L2:int[] -> L3:int[])
   // L3 enthaelt abwechselnd ein Element aus L1 gefolgt von einem Element
   // aus L2, solange das moegliche ist, und dann
   // die "ueberzaehligen Elemente" der laengeren in-Liste.
   // Beispiel: Nach dem Aufruf
   // aus2mach1(int[1,2], int[3,4,5,6] -> L3)
   // ist L3 gleich int[1,3,2,4,5,6]
```

Aufgabe 4 (20 Punkte):

Nehmen Sie an:

Sie haben eine **Z-Maschine**,
die beiden Compiler **C1** und **C2** und
den Interpreter **I1.**
Wie können Sie damit den Compiler **C4** erzeugen?

Geben Sie als Lösung entsprechende TI-Diagramme an.



Beurteilung der Klausur:

Punkte↓	
Aufgabe 1:	Note:
Aufgabe 2:	Datum:
Aufgabe 3:	
Aufgabe 4:	
Summe:	

Korrigierte Beurteilung der Klausur:

Punkte↓	
Aufgabe 1:	Note:
Aufgabe 2:	Datum:
Aufgabe 3:	
Aufgabe 4:	
Summe:	

Lösung 1 (20 Punkte): Geben Sie eine (kontextfreie, Typ-2-) Grammatik an für die Menge aller natürlichen Zahlen (0, 1, 2, ...) im **3-er-System**, die (ohne Rest) **durch 5 teilbar** sind. Startsymbol `RK0` (wie "Restklasse 0").

```
R01: RK0 : "0"          R07: RK3 : RK1 "0"      R13: RK4 : RK3 "0"
R02: RK1 : "1"          R08: RK4 : RK1 "1"      R14: RK0 : RK3 "1"
R03: RK2 : "2"          R09: RK0 : RK1 "2"      R15: RK1 : RK3 "2"

R04: RK0 : RK0 "0"      R10: RK1 : RK2 "0"      R16: RK2 : RK4 "0"
R05: RK1 : RK0 "1"      R11: RK2 : RK2 "1"      R17: RK3 : RK4 "1"
R06: RK2 : RK0 "2"      R12: RK3 : RK2 "2"      R18: RK4 : RK4 "2"
```

Lösung 2 (20 Punkte): Betrachten Sie die folgende Grammatik für Ausdrücke (Endsymbole sind in doppelte Anführungszeichen eingefaßt, z.B. `"#1"` oder `"c"`. `AUS1` ist das Startsymbol):

```
AUS1 : AUS2 "#1" AUS1
AUS1 : AUS2

AUS2 : AUS2 "#2" AUS3
AUS2 : AUS3

AUS3 : "#3" AUS3
AUS3 : AUS4

AUS4 : "#4" AUS5
AUS4 : AUS5

AUS5 : LIT
AUS5 : "(" AUS1 ")"

LIT  : "a"
LIT  : "b"
LIT  : "c"
```

In dieser Grammatik kommen die Operatoren `#1`, `#2`, `#3`, `#4` vor. Geben Sie als Antworten auf die folgenden Fragen die zutreffenden Operatoren an:

2.01.	Alle zweistelligen Operatoren?	**#1, #2**
2.02.	Alle einstelligen Operatoren?	**#3, #4**
2.03.	Alle linksassoziativen Operatoren?	**#2**
2.04.	Alle rechtsassoziativen Operatoren?	**#1**
2.05.	Welcher Operator bindet am stärksten?	**#4**
2.06.	Welcher Operator bindet am schwächsten?	**#1**

Welche der folgenden Worte kan man aus obiger Grammatik ableiten (JA) und welche nicht (NEIN)?

2.07.	`#3 #3 a`	**JA**
2.08.	`#4 #4 b`	**NEIN**
2.09.	`a #2 b #1 c`	**JA**
2.10.	`c #1 c #2 a`	**JA**

Lösung 3 (40 Punkte):

3.1. Schreiben Sie ein Prädikat entsprechend der folgenden Spezifikation:

```
proc listeDerBesonderen(L:int[] -> BL:int[])
   // Die Liste BL enthaelt alle besonderen Elemente der Liste B.
   // Welche Elemente "besonders" sind, entscheidet das Praedikat
   //
   // condition istBesonders(N:int)
   //    // Gelingt, wenn die Zahl N besonders ist
   //
   // Nehmen Sie an, dass das Praedikat istBesonders vorgegeben ist.

   rule listeDerBesonderen(int[] -> int[]):
   rule listeDerBesonderen(int[N::R] -> ERG):
      listeDerBesonderen(R -> BR)
      {
         istBesonders(N)
         ERG <- int[N::BR]
      |
         ERG <- BR
      }
```

3.2. Schreiben Sie ein Prädikat entsprechend der folgenden Spezifikation:

```
condition keineZwillinge(L:int[])
   // Gelingt genau dann wenn L keine Zwillingspaare enthaelt. Ein Zwillingspaar
   // besteht aus zwei gleichen, benchbarten Elementen.
   // Beispiele:
   // Die Liste int[1, 2, 1, 2] enthaelt keinen Zwilling
   // Die Liste int[1, 2, 2, 1] enthaelt einen Zwilling

   rule keineZwillinge(int[]):
   rule keineZwillinge(int[N]):
   rule keineZwillinge(int[N1,N2::R]):
      Unequal(N1, N2)
      keineZwillinge(int[N2::R])
```

3.3. Schreiben Sie ein Prädikat entsprechend der folgenden Spezifikation:

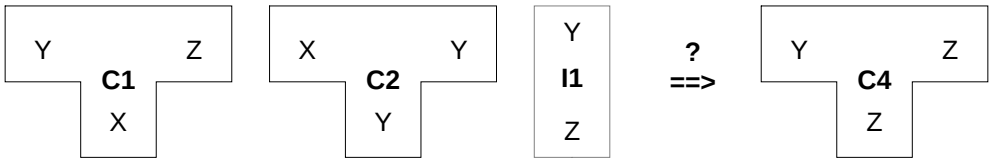
```
proc aus2mach1(L1:int[], L2:int[] -> L3:int[])
   // L3 enthaelt abwechselnd ein Element aus L1 gefolgt von einem Element
   // aus L2, solange das moegliche ist, und dann
   // die "ueberzaehligen Elemente" der laengeren in-Liste.
   // Beispiel: Nach dem Aufruf
   // aus2mach1(int[1,2], int[3,4,5,6] -> L3)
   // ist L3 gleich int[1,3,2,4,5,6]

   rule aus2mach1(int[], L2 -> L2):
   rule aus2mach1(L1, int[] -> L1):
   rule aus2mach1(int[N1::R1], int[N2::R2] -> int[N1, N2::L12]):
      aus2mach1(R1, R2 -> L12)
```

Lösung 4 (20 Punkte):

Nehmen Sie an:

Sie haben eine **Z-Maschine**,
die beiden Compiler **C1** und **C2** und
den Interpreter **I1.**
Wie können Sie damit den Compiler **C4** erzeugen?

Geben Sie als Lösung entsprechende TI-Diagramme an.



Lösung: