Aufgabe 1 Unifikation (15 Punkte)

Testen Sie die folgenden Termpaare auf Unifizierbarkeit. Geben Sie. falls vorhanden, den allgemeinsten Unifikator an bzw. begründen Sie, warum kein solcher existiert. (Dabei sind x, y, z Variablen, die anderen Buchstaben sind Konstanten bzw. Funktionssymbole.)

Teil 1 (3 Pkt.):
$h(r(a), l(z), g(g(y)))$ und $h(x, y, x)$

Teil 2 (3 Pkt.):
$h(r(a), l(z), g(g(y)))$ und $h(x, y, z)$

Teil 3 (3 Pkt.):
$h(r(a), l(x), g(g(y)))$ und $h(x, y, z)$

Teil 4 (3 Pkt.):
$3 * 5$ und 7

Teil 5 (3 Pkt.):
$3 * 5$ und $5 * 3$

Aufgabe 2 Dreierliste (12 Punkte)

Implernentieren Sie ein Prädikat *dreierliste*/2, das genau dann erfiillt ist, wenn das erste Argument eine Liste ist und das zweite Argument, eine Liste von dreielementigen Listen, die konkateniert die erste Liste ergeben; die letzte Unterliste des zweiten Arguments darf auch 1 oder 2 Elemente enthalten. Die leere Liste kommt als Argument nicht vor.

Beispiel:

```
| ? - dreierliste([1,2,3,4,5,6],X).
X = [[1,2,3],[4,5,6]] ? ;
no

| ? - dreierliste([1,2,3,4],X).
X = [[1,2,3],[4]] ? ;
no
```

Aufgabe 3 Programmausführung in Prolog (10 Punkte)

Betrachten Sie das folgende Prolog-Programm:

```
pfeil(a,b).
pfeil(a,c).
pfeil(b,d).
pfeil(c,d).
pfeil(c,e).
weg(X,Y) :- pfeil(X,Y).
weg(X,Y) :- pfeil(X,Z), weg(Z,Y).
```

Teil 1 (5 Pkt.):
Geben Sie alle vom System generierten Lösungen für die Anfrage ?- **weg** (a, X) . in der Reihenfolge ihres Auftretens an.

Teil 2 (5 Pkt.):
Wie ändert sich das in Teil 1 angegebene Verhalten, wenn die letzte Programmzeile ersetzt, wird durch

```
weg(X, Y) :- pfeil(X,Z),!, weg(Z,Y).
```

Geben Sie für das so geänderte Programm wiederum alle vom System generierten Lösungen für die Anfrage ?- `weg(a,X)` . in der Reihenfolge ihres Auftretens an.

Aufgabe 4 Zahlensysteme (13 Punkte)

Eine natürliche Zahl z lässt sich wie folgt in ein Zahlensystem mit Basis $b \in \{2, 3, \ldots\}$ umrechnen:'

- Das Ergebnis für $z = 0$ ist 0 (bzw. die Liste mit dem einzigen Element 0), unabhängig von b.
- Für $z \neq 0$ dividiere z durch b mit ganzzahligem Ergebnis z' und Rest r. Die Berechnung wird für z' wiederholt, und an das Ergebnis wird eine Stelle mit dem Wert r *hinten* angefügt.

Implernentieren Sie eine Scheme-Funktion *(basiswechsel zahl basis)* die eine Zahl *zahl* und eine Basis *basis* erwartet und die Repräsentation von Zahl zur Basis *basis* zurückliefert.

Beispiel:

```
> (basiswechsel 10 2)
(0 1 0 1 0)
> (basiswechsel 0 3)
(0)
> (basiswechsel 60000 16)
(0 14 10 6 0)
```

Hinweis: Die vordefinierte Funktion für die Ganzzahldivision heißt *quotient*, die für den Divisionsrest *remainder*. Weitere aus dem Kurs bekannte Funktionen n-ie z.B. *uppend* können Sie ebenfalls als gegeben voraussetzen.

[1] Zur Vereinfachung erfolgt die Darstellung der Ergebnisse in Listen mit jeweils einem Integereintrag für jede Stelle und einer führenden 0.

Aufgabe 5 Listenordnung (20 Punkte)

Sei E eine Menge mit den Relationen $=_E$ und $<_E$. Dann definieren wie eine Relation $<_L$ auf Listen mit Elementen von E wie folgt:

Seien $l_1 = (e_{1,1}\ e_{1,2}\ \ldots\ e_{1,k})$ und $l_2 = (e_{2,1}\ e_{2.2}\ \ldots\ e_{2,l})$ Listen mit Elementen $e_{i,j} \in E$. Es gilt $l_1 <_L l_2$ genau dann, wenn eine der folgenden Bedingungen gilt:

- $l > k$ und $e_{1,i} =_E e_{2,i}$ für alle $i \in \{1, \ldots, k\}$. D.h. l_1 ist kürzer als l_2 und bis zur Länge von l_1 sind die Elemente identisch.
- Es gibt ein n mit k > n, 1 > n, e $_{1,i} =_E e_{2,i}$ für alle $* \in \{1, \ldots, n\}$ und $e_{1,n+1} <_E e_{2,n+1}$, d.h. die ersten n Elemente sind gleich und das ▪ + 1-te Element von l_1 ist kleiner als das von l_2.

Implernentieren Sie eine Funktion (*l-kleiner* l_1 l_2 *e-gleich* e-kleiner) die folgende Eingaben erwartet:

- l_1 und l_2 sind Listen mit Elementtyp E.
- Die Funktion *e-gleich* erwartet zwei Elemente er, $e_2 \in E$ als Eingabe und liefert `#t` falls $e_1 =_E e_2$, ansonsten `#f`.
- Die Funktion e-kleiner erwartet zwei Elemente e_1, $e_2 \in E$ als Eingabe und liefert `#t` falls $e_1 <_E e_2$, ansonsten `#f`.

l-kleiner liefert `#t`, falls $l_1 <_L l_2$ und #f sonst.

Beispiel:

```
> (l-kleiner '(1 2 3) ' (1 2 4) = <)
#t
> (l-kleiner '(1 2) '(1 2 3) = <)
#t
> (l-kleiner '(1 2 4) '(1 2 3) = <)
#f
> (l-kleiner '(1 2 3)'(1 2) = <)
#f
```

Aufgabe 6 Binärbäume (10 Punkte)

Binärbäume lassen sich in Scheme als dreielementige Listen darstellen: wobei das erste Element' den Wert des Wurzelknotens enthält, das zweite Element den linken Teilbaum und das dritte Element den rechten Teilbaum.

Teil 1 (5 Pkt.):
Definieren Sie eine Funktion $(mk\text{-}tree\ v\ l\ r)$, die einen Wert v und Bäume l und r erwartet und einen Baum b mit Wurzelwert v, rechtem Teilbaum r und linkem Teilbaum l zurückliefert.

Teil 2 (5 Pkt.):
Lässt sich in analoger Weise eine Funktion $(mk\text{-}inf\text{-}tree\ v\ l\ r)$ definieren. deren Teilbaume unendlich sein können? Begründen Sie Ihre Antwort.

Aufgabe 7 Lösungssuche (20 Punkte)

Jeder der drei Freunde Anton, Hans und Karl spielt zwei der folgenden Musikinstrumente: Bass, Keyboard, Schlagzeug, Gitarre, Trompete, Piano. Der Schlagzeuger verulkt den Bassisten gern. Der Trompeter und der Schlagzeuger gehen mit Anton gelegentlich zum Fußball. Der Gitarrist hat beim Keyboarder Schulden. Der Bassist ist mit der Schwester des Gitarristen verlobt. Hans hat dem Trompeter das Instrument, versteckt. Beim Skat hat Karl gegen Hans und den Gitarristen gewonnen. Wer spielt welche Instrumente?

Implernentieren Sie mittels constraintlogischer Programmierung ein Prädikat *musiker/1*, welches das gegebene Problem löst, d.h. das genau dann erfüllt ist. wenn das Argument eine Liste ist in der die Werte 1 für Karl, 2 für Anton und 3 für Hans jeweils zweimal vorkommen und zwar an den Positionen der jeweiligen Instrumente. Die Instrumente haben dabei die Reihenfolge

1. *Bass*
2. *Schlagzeug*
3. *Gitarre*
4. *Keyboard*
5. *Trompete*
6. *Piano*

Hinweise:

1. Eine Textaussage wie “Der Schlagzeuger verulkt den Bassisten gern” ist so zu deuten. dass der Schlagzeuger und der Bassist unterschiedliche Personen sind.
2. Bei Ihrer Lösung dürfen Sie bekannte Prädikate und Funktionen (in Prolog/CLP(FD) z. B. *member, permutation, all_different* etc.) als gegeben voraussetzen.