Vorlesung Funktionale Programmierung, *WS 02/03*	*Dozentin: Dr. Ute Schmid,* *Tutor: Jens Waltermann*

Übungsblatt 5

(Ausgabe: Mo. 11.11.02, Abgabe: Fr. 15.11.02 12.00)

1. Aufgabe Unify:

   Zu schreiben ist diesmal eine Funktion `unify` zur Unifikation von Typtermen, Grundvoraussetzung um einen Typcheck zu implementieren. Folgende Vorgaben werden Ihnen gemacht:

   - Ein Datentyp `typterm`, der alle erlaubten Typterme definiert, wie sie in den zu prüfenden Ausdrücken vorkommen können.
     ```
     datatype typterm =  INT | REAL | BOOL | STRING |
                         --> of typeterm * typeterm |
                         ** of typeterm * typeterm |
                         V of int;
     ```
     V steht in diesem Fall für Variablen.
   - Die Operatoren `-->` und `**` sollen als Infixoperatoren benutzt werden:
     ```
     infix -->;
     infix **;
     ```
   - Die Funtion `unify` soll vom Typ:
     `(typeterm * typeterm) -> (typeterm * ((typeterm * typeterm) list)`
     sein. Dabei soll `unify` prüfen, ob die beiden eingegebenen Typterme unifizierbar sind. Falls ja, soll der unifizierte Typterm und die Liste von möglichen Variablenbindungen geliefert werden. Beispiel:
     `unify((INT ** V(1)) --> V(2), V(3) --> BOOL)`
     wertet aus zu:
     `((INT ** V(1)) --> BOOL, [(V(3), INT ** V(1)), (V(2), BOOL)])`
   - Andernfalls soll eine Exception auftreten. Sie erschaffen diese Exception indem Sie :
     `exception FAIL;`
     eingeben. Ausgelöst wird diese Exception – analog zur `throw`-Anweisung in Java – durch den Befehl:
     `raise FAIL;`
     Ein Beispiel für eine fehlerhafte Eingabe:
     `unify(BOOL ** V(1), INT ** INT)`
     wertet aus zu
     `FAIL`

2. Aufgabe(*) Finden Sie Beispiele, bei denen dieser Algorithmus versagt.Und erläutern Sie, warum.