

1 Haskell (Datenstruktur)

Wir wollen uns mit der **Union-Find**-Datenstruktur beschäftigen. Diese Datenstruktur wird unter anderem verwendet um den Algorithmus von Kruskal effizient implementieren zu können. Im Allgemeinen verwendet man dazu jedoch Mengen von Bäumen anstatt Listen. Der Einfachheit wegen wollen wir die Datenstruktur aber via Listen implementieren. Als erstes wird die Datenstruktur mit der `prepare`-Funktion und einer Menge S initialisiert.

```
prepare :: [a] -> [[a]]
prepare [x] = [[x]]
prepare (x:xs) = [[x]] ++ prepare xs
```

Nach der Initialisierung erhalten wir eine Liste von Listen. Jede der inneren Listen stellt eine Komponente dar. Nun soll die `union`-Funktion eine Datenstruktur und zwei Elemente x und y erhalten. Ihre Rückgabewert ist eine neue Datenstruktur, in der die Komponenten der beiden Elemente verschmolzen wurden. Hier ein Beispiel:

```
union [[1],[2],[3],[4]] 1 3 -> [[1,3],[2],[4]]
```

Die Funktion `find` erhält eine Datenstruktur und ein Element. Ihre Aufgabe ist es einen Vertreter aus der Komponente in der sich das Element befindet zurückzugeben. Ein Beispiel:

```
find [[1,2,3],[4,5],[6]] 1 -> 1
find [[1,2,3],[4,5],[6]] 3 -> 1
```

Zu guter letzt soll die Funktion `same` überprüfen, ob sich zwei Elemente in der selben Komponente befinden. Dazu bekommt sie eine Datenstruktur und die beiden Elemente übergeben. Ein Beispiel:

```
same [[1,2],[3,4],[5]] 3 4 -> True
same [[1,2],[3,4],[5]] 1 5 -> False
```

Deine Aufgabe ist es nun, die Funktionen `union`, `find` und `same` zu implementieren. Dabei solltest du sinnvolle Annahmen über die verwendeten Typklassen treffen.

2 Haskell (Funktionstypen)

Betrachte die folgenden Funktion und gib ihren Typen an sofern dieser gültig ist. Andernfalls erkläre, warum die Funktion keinen gültigen Typen hat. Ziehe falls relevant auch die Typklassen `Eq` und `Ord` in Betracht.

```
f x y z = f (f y x z) x y++z

g f x y = x ++ map (f x y) (x:y:[[23]])

h f x = f h x
```

3 Prolog

Wir befinden uns im alten Japan in dem Dorf Osaka, welches von den Samurai Kiyoshi, Ryo, Shin und Akiyama bewohnt wird. Die Samurai sind zudem stolze Väter. Kiyoshi ist der Vater von Fudo. Ryoichi ist der erste Sohn des Ryo, Shinichi der Erstgeborene des Shin. Akiyama nennt seinen Sohn Taro. Ryoichi hat ebenfalls einen Sohn namens Mamoru. Modelliere die Verwandtschaft durch Fakten der Form `vater(X,Y)`, falls X der Vater von Y ist.

Eines Tages ermordet Kiyoshi Ryo wegen eines Streits. Der Tradition entsprechend muss der erstgeborene Sohn seinen Vater rächen, indem er dessen Mörder tötet. Modelliere den ersten Mord in einem Faktum und eine Regel `mord(X,Y)` um die blutigen Konsequenzen dieser Tradition zu modellieren. Modelliere zudem ein Prädikat `leben(X)` das beschreibt, welche Samurai überleben.

Die Mädchen Mamiko, Midori, Kameko und Yumi müssen mit ansehen, wie sich die Samurai reihenweise das Leben nehmen. Dabei sind die Mädchen mit einigen der Männer intim involviert. Die junge Mamiko ist an Fudo und Midori an Ryoichi interessiert. Kamekos Herz gehört Mamoru während Taro Yumi verzaubert hat. Modelliere das Interesse der Mädchen an den entsprechenden Samurai indem du neue Fakten der Form `liebt(X,Y)` einführst.

Dummerweise hat das jüngst ausgebrochene Blutbad die Hochzeitspläne der Mädchen durcheinander geworfen, schliesslich kann man keinen Toten heiraten. Die betroffenen Mädchen trauern nun um ihre geliebten Männer. Modelliere ein Prädikat `trauer(X)`, welches angibt ob ein Mädchen trauert.

Den Mädchen wird bewusst, dass die Bevölkerung des Dorfes wieder vergrößert werden muss. Demnach erklären sich die Mädchen, deren wahre Liebe verstorben ist, dazu bereit einen Mann zu heiraten den sie nicht lieben. Allerdings würde kein Mädchen jemals den Mann heiraten, der ihren Geliebten getötet hat. Dazu kommt, dass sich die Mädchen versprochen haben keine bestehenden Liebespläne zu durchkreuzen. Modelliere eine Regel `heiratetNicht(X,Y)`, die angibt ob das Mädchen X den Samurai Y heiraten kann. Beachte dabei alle Bedingungen! Stelle Anfragen um herauszufinden, welche möglichen Ehen geschlossen werden können.

Zeichne für die Anfrage `heiratetNicht(Kameko,X)` den (ebenfalls) traditionellen SLD-Baum.

4 Prolog (Geplänkel)

Was kann der Grund für das Fehlschlagen der Unifikation zweier Terme sein. Benenne und beschreibe die möglichen Fehler. Unifiziere wahllos zusammengesetzte Terme als Vorbereitung für die Klausur.

5 Prolog (Listen)

Denke dir eine möglichst komplizierte Aufgabe aus in der Listen mehr als nur marginal involviert sind. Löse dann diese Aufgabe oder tausche sie mit deinen Freunden oder was weiss ich.