2. Aufgabenblatt zu Funktionale Programmierung vom 23.10.2006.
Fällig: 30.10.2006 / 06.11.2006 (jeweils 15:00 Uhr)

Themen: *Funktionen auf Zeichenreihen, ganzen Zahlen und Listen*

Für dieses Aufgabenblatt sollen Sie die zur Lösung der unten angegebenen Aufgabenstellungen zu entwickelnden Haskell-Rechenvorschriften in einer Datei namens `Aufgabe2.lhs` in Ihrem home-Verzeichnis ablegen. **Anders** als bei der Lösung zum ersten Aufgabenblatt sollen Sie also dieses Mal ein "**literate Script**" schreiben.

Versehen Sie wie schon in Ihrer Lösung zum ersten Aufgabenblatt auch hier alle Funktionen, die Sie zur Lösung benötigen, mit ihren Signaturen und kommentieren Sie Ihre Programme wieder aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Im einzelnen sollen Sie folgende Problemstellungen bearbeiten:

1. In dieser und der folgenden Aufgabe betrachten wir noch einmal die Realisierung eines einfachen Editors in Haskell mittels des Typs:

   ```
   type Editor = [Char]
   ```

   Schreiben Sie eine Haskell-Rechenvorschrift `maxLgthCommonSubstr` mit der Signatur `maxLgthCommonSubstr :: Editor -> Editor -> Int`, die angewendet auf zwei Editoren `e1` und `e2` die Länge des längsten gemeinsamen Teilausdrucks von `e1` und `e2` bestimmt. Der Aufruf von `maxLgthCommonSubstr "Donaudampfschiff" "Modellschiffbau"` soll also das Resultat `6` liefern.

2. Schreiben Sie eine Haskell-Rechenvorschrift `maxCommonSubstr` mit der Signatur `maxCommonSubstr :: Editor -> Editor -> ([String],Int)`, die angewendet auf zwei Editoren `e1` und `e2` ein Paar aus der alphabetisch geordneten Liste der gemeinsamen Teilausdrücke maximaler Länge von `e1` und `e2` und der Länge dieser Teilausdrücke liefert.

   Bei der Implementierung können Sie davon ausgehen, dass die Funktion `maxCommonSubstr` nur für Editoren aufgerufen wird, die ausschließlich Klein- und Großbuchstaben enthalten; Sonderzeichen, Ziffern, Leerstellen, etc. kommen nicht vor. Der Aufruf von `maxCommonSubstr "Autorennbahnwerkstatt" "Eisenbahnergewerkschaft"` soll also das Resultat `([nbahn","werks"],5)` liefern.

3. Eine Zeichenreihe `s` heißt Palindrom, wenn `s` sich in gleicher Weise von links nach rechts wie von rechts nach links liest. So ist z.B. die Zeichenreihe `otto` ein Palindrom der Länge 4 (die Zeichenreihe `Otto` ist kein Palindrom!). Schreiben Sie eine Haskell-Rechenvorschrift `maxLgthPal` mit der Signatur `maxLgthPal :: String -> Int`, die angesetzt auf eine Zeichenreihe `t` die Länge des längsten in `t` als Teilzeichenreihe vorkommenden Palindroms bestimmt. Der Aufruf `maxLgthPal "Ottos Lebensmotto"` soll also das Resultat 4 liefern.

4. Eine Relation $R \subseteq IN \times IN$ (über den natürlichen Zahlen) heißt *transitiv*, wenn folgendes gilt:

$$\forall\, x, y, z \in IN.\ x\, R\, y\ \wedge y\, R\, z\ \Rightarrow\ x\, R\, z$$

Konzeptuell lassen sich Relationen über IN als quadratische Matrizen über Wahrheitswerten auffassen. Dabei steht an der Position (m, n) der Wahrheitswert `wahr` genau dann, wenn m und n in Relation R zueinander stehen, d.h. wenn $m\, R\, n$ gilt.

In Haskell lässt sich dies wie folgt realisieren:

```
type Relation = [[Bool]]
```

Dabei steht die m-te Teilliste für die m-te Zeile der Matrix, die n-te Spalte der Matrix ergibt sich aus der Projektion der Matrixzeilen auf ihre n-ten Komponenten.

Schreiben Sie nun eine Wahrheitswertfunktion `istTransitiv` mit der Funktionalität `istTransitiv :: Relation -> Bool`, die angewendet auf eine Relation `r` den Wahrheitswert `True` liefert, falls `r` transitiv ist, ansonsten den Wahrheitswert `False`. Ist die Argumentmatrix nicht quadratisch, d.h. sind nicht alle Zeilen und Spalten von gleicher Länge, so liefert die Funktion `istTransitiv` den Wahrheitswert `False`.

Denken Sie bitte daran, dass Sie für die Lösung dieses Aufgabenblatts ein “literate” Haskell-Skript schreiben sollen!