Universität Tübingen, Wilhelm-Schickard-Institut, WS 2018/19

Spezielle Kapitel der Praktischen Informatik: Compilerbau (INF3199A)

Übungsblatt 1, Besprechungstermin: 8. November 2018

1.1 Logische Operationen

a) Geben Sie in Haskell Definitionen für folgende logische Operatoren an:

```
and, or, xor :: (Bool, Bool) -> Bool
neg :: Bool -> Bool
```

b) Programmieren Sie mithilfe der Operatoren Funktionen, die einem Halb– bzw. Volladdierer entsprechen.

1.2 Symbolisches Differenzieren

a) Schreiben Sie einen Datentyp `Term` der (gebrochen)rationale Funktionen in einer Variablen repräsentiert.

   Hinweis: Sie benötigen jeweils einen Konstruktor für *Monome*, *Addition*, *Multiplikation* und *Division*.

b) Programmieren Sie eine Funktion `diff: Term -> Term`, welche die erste Ableitung bestimmt.

   Hinweis: Sie benötigen für jeden Konstruktor des Typs `Term` eine Gleichung für die Funktion `diff`.

1.3 `curry3`, `uncurry3`

Schreiben Sie die Funktionen

```
curry3 :: ((a, b, c) -> d) -> (a -> b -> c -> d)
uncurry3 :: (a -> b -> c -> d) -> ((a, b, c) -> d)
```

die die jeweiligen Funktionen mit dem Tupel Typ–Konstruktor in äquivalente Funktionen mit dem `->` Typ–Konstruktor umwandeln und umgekehrt.

1.4 Die Funktion `map`

Schreiben Sie eine Funktion, `toListofLists :: [a] -> [[a]]`, die jedes Element einer Liste zu einer einelementigen Liste macht. Nutzen Sie dazu die Funktion `map`.

1.5 Die Funktionen `foldl` und `foldr`

Programmieren Sie eine Funktion `listStringConcat :: [String] -> String`, die alle Elemente einer `String`–Liste aneinander hängt.

Nutzen Sie dazu entweder die Funktion `foldl` oder `foldr`.

Erhalten Sie unterschiedliche Ergebnisse?

1.6 Abstrakter Datentyp `BBaum`

a) Programmieren Sie den polymorphen abstrakten Datentyp `BBaum a`. `BBaum a` steht für einen Binärbaum und soll zwei Konstruktoren enthalten. Einen Konstruktor für einen leeren Binärbaum und einen Konstruktor der zwei Teilbäume und ein Element zu einem neuen Binärbaum zusammenfügt.

b) Schreiben Sie eine Funktion `mapTree :: BBaum a -> (a -> b) -> BBaum b`. Der Funktionsaufruf `mapTree b f` soll auf jedes Element des Eingabebaums `b` die Funktion `f` anwenden.