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

Spezielle Kapitel der Praktischen Informatik: Compilerbau (INF3199A)

Übungsblatt 2, Besprechungstermin: 15. November 2018

2.1 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`.

2.2 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?

2.3 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.

2.4 Java–Scanner

Schreiben Sie mit Alex für einen Ausschnitt aus Java einen Scanner.

Dazu müssen Sie eine Lex–Spezifikation der Form erstellen:

```
{
...
}

%wrapper ...

tokens :-

  abstract {}
  boolean {}
  break {}
```

```
  case {}

  ...

  [1-9][0-9]*[L]?      { }
  0[xX][0-9a-fA-F]+[L]? { }

{

data Token
     = ABSTRACT
     | BOOLEAN
     | BREAK
     | CASE
     | CATCH
...
     | INTLITERAL Integer
     | BOOLLITERAL Bool
     | JNULL
     | CHARLITERAL Char
     | STRINGLITERAL String
     | IDENTIFIER String
...

}
```