Prof. Dr. R. Loogen, M. Dieterle
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
D-35032 Marburg

18. Oktober 2011

Philipps Universität Marburg

Übungen zu „Grundlagen des Compilerbau“, Winter 2011/12

Nr. 1, Abgabe der Aufgaben: 25. Oktober 2011 vor der Vorlesung

---

Hinweise:

- Die Übungsblätter erscheinen dienstags, sind am darauffolgenden Dienstag abzugeben und werden bei Rückgabe im Tutorium besprochen.
- Bei Programmieraufgaben sind die Programme bitte *zusätzlich zur Papierversion* per Mail an `dieterle @ mathematik` abzugeben. Ihre Programme müssen kompilierbar sein, um bepunktet zu werden.
- Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.
- Die Präsentation von zwei Aufgaben im Tutorium ist Bedingung für den Erwerb eines Leistungsnachweises. Bitte achten Sie selbst (mit) darauf, dass Sie diese Bedingung erfüllen können.
- Wir arbeiten mit der Programmiersprache Haskell. Einen kompakten Überblick über Haskell-Funktionen und Datentypen finden Sie unter:
  http://zvon.org/other/haskell/Outputglobal/index.html.
  Besonders hilfreich beim nachschlagen von Funktionen ist die Haskell API Suche Hoogλe:
  http://haskell.org/hoogle.
  Das Buch *Real World Haskell* bietet einen guten Einstieg und steht sowohl in unserer Bibliothek als auch komplett Online zur Verfügung:
  http://book.realworldhaskell.org.

Aufgaben

1.1 **Bootstrapping** — 4 Punkte

Sie haben die Aufgabe, einen Java-Übersetzer zu entwickeln, der auf einer SGI-Workstation läuft und Code für die Sony Playstation 3 erzeugt. Ihr Auftraggeber verlangt, dass der Übersetzer in Java geschrieben werden soll. Es soll am Ende sowohl der Quellcode des Übersetzers in der Sprache Java, als auch der compilierte Übersetzer in SGI-Maschinensprache präsentiert werden.
Ihnen stehen zwei auf SGI lauffähige C-Übersetzer zur Verfügung, welche Maschinencode für SGI bzw. für die Playstation erzeugen. Geben Sie in Form von T-Diagrammen an, wie Sie (**mit möglichst wenig Programmieraufwand**) vorgehen können und an welchen Stellen “Handarbeit” erforderlich ist.

Bitte wenden!

1.2 Entfernung von Whitespace

3 Punkte

Schreiben Sie ein Haskell-Programm, das aus einer gegebenen Zeichenkette Reihen von Leerzeichen, Tabulatoren (`\t`) und Zeilenumbrüchen (`\n`) entfernt und durch jeweils ein Leerzeichen ersetzt.

(a) Schreiben Sie zunächst eine Funktion `isWhiteSpace :: Char → Bool` die erkennt, ob der Argumentbuchstabe zu den oben genannten Zeichen gehört.

(b) Schreiben Sie nun eine Funktion `compress :: [Char] → [Char]` die unter Benutzung von `isWhiteSpace` Ketten von mehr als einem Leerzeichen zu einem einzigen Leerzeichen reduziert.

Beispiel:

```
"Emil und   die Detektive\n\n \t\t von Erich Kaestner    "
```

wird konvertiert zu

```
"Emil und die Detektive von Erich Kaestner "
```

1.3 Scannen eines Strings

5 Punkte

(a) Definieren Sie eine Haskell-Funktion `getStrings :: String → [String]`, die einen `String` erhält und alle zusammenhängenden Zeichenketten extrahiert, welche nur aus Kleinbuchstaben, Großbuchstaben, Zahlen, Leerzeichen und dem '_' - Symbol bestehen. Sie können dazu die Funktionen:

```
isLower, isUpper, isDigit, isSpace :: Char → Bool
```

aus dem Modul `Data.Char` benutzen.

(b) Schreiben Sie ein Haskell-Programm, das die Funktion `getStrings` auf die kompilierte Version des eigenen Codes anwendet. Die von `getStrings` erstellte Wortliste soll ausgegeben werden. Kompilieren Sie das Programm zu Testzwecken mit "ghc". Nützliche Funktionen sind:

```
getProgName :: IO String
readFile    :: String → IO String

putStrings :: [String] → IO [()]
putStrings = mapM putStrLn
```