PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen
J.Berthold

D-35032 Marburg
Hans Meerwein Straße

27.Oktober 2005

1. Übung zu „Grundlagen des Compilerbaus“, WS 2005/06

Abgabe schriftlicher Aufgaben:	**Do, 3.November 2005 (vor der Vorlesung)**
Besprechung mündlicher Aufg.:	**ab 31.Oktober 2005 in der Übung**

Hinweise:

- Die Lösungen zu schriftlichen Aufgaben sollen grundsätzlich schriftlich abgegeben, Programme ausgedruckt und *zusätzlich* per E-Mail an den jeweiligen Tutor geschickt werden. Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt.
- Mündliche Aufgaben sollen zum angegebenen Tutoriumstermin vorbereitet werden, sie werden im Tutorium präsentiert und besprochen.
- Die Präsentation von mündlichen und schriftlichen Aufgaben im Tutorium ist Bedingung für den Erwerb eines Leistungsnachweises. Bitte achten Sie selbst (mit) darauf, dass Sie diese Bedingung erfüllen können.

Mündliche Aufgaben

1.1 **T-Diagramme**
Gegeben sei ein C-Compiler, der auf einer Rechnerplattform M läuft und Maschinensprache für M generiert, sowie ein in C geschriebenes Java-nach-C-Übersetzungsprogramm. Benutzen Sie die in der Vorlesung vorgestellten Diagramme, um die folgenden Vorgänge darzustellen:

(a) Kompilation des Java-nach-C-Übersetzers in Maschinensprache

(b) Entwicklung eines Java-nach-M-Übersetzers in Maschinensprache

1.2 **Entfernung von Whitespace**
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 "`

Bitte wenden!

Schriftliche Aufgaben

1.3 Konvertierungsfunktionen für römische Zahlen 8 Punkte

Die römische Zahlschreibweise benutzt bekanntlich die Grundzeichen `I,V,X,L,C,D` und `M` mit Werten entsprechend der Tabelle. Andere Zahlen werden als Verkettung dieser Grundzeichen nach den angegebenen Regeln ausgedrückt.

Regeln:

Die Zeichen `I,X,C,M` kommen höchstens dreimal *hintereinander* vor, die Zeichen `V,L,D` insgesamt nur einmal.
Steht *rechts* von einem Zeichen ein Zeichen mit gleichem oder kleinerem Wert, so wird sein Wert addiert.
Nur die Zeichen `I,X` und `C` dürfen *links* neben einem Zeichen mit größerem Wert stehen, und zwar *höchstens einmal*. In diesem Fall wird der kleinere Wert vom größeren subtrahiert. Die Werte der Zeichen rechts vom Zeichen mit dem größerem Wert dürfen höchstens den Wert des kleineren Zeichens haben, das kleinere Zeichen darf aber nicht direkt wieder auf das größere folgen.

Grundzeichen:

1	I
5	V
10	X
50	L
100	C
500	D
1000	M

Beispiele: `LXXXIX`$= 89$ `CDIL`$= 449$ `ID`$= 499 =$`CMXCIX` `MCMLXXVIII`$= 1978$

Ungültig sind z.B. `XXXX,VC,XML`

(a) Definieren Sie eine Haskell-Funktion `fromRoman :: String -> Int`, / 3
die korrekte römische Zahlen in Dezimalzahlen umwandelt.

(b) Schreiben Sie eine Umkehrfunktion `toRoman :: Int -> String`, / 3
die Zahlen aus dem Dezimalsystem in korrekte römische Zahlen umwandelt.
Es soll gelten: $n =$ `fromRoman ( toRoman` n `)` $\forall n \in [1, 3999]$.

(c) Gilt auch `s = toRoman ( fromRoman s)` für gültige römische Zahlen s? / 2
Sorgen Sie dafür, dass ungültige Strings eine verständliche Fehlermeldung erzeugen.

1.4 GNU Compiler Collection (GCC) 4 Punkte

In GNU-Compilern wird eine plattformunabhängige Maschinensprache namens RTL als Zwischensprache verwendet. Es gibt Übersetzer, die in Hochsprachen wie `C, C++` oder Pascal geschriebene Programme nach RTL übersetzen; analog gibt es Übersetzer, die RTL-Programme auf Plattformen wie z.B. PPC, x86 oder IA-64 spezialisieren. Außerdem gibt es einen RTL-Optimierer, der ein RTL-Programm in ein effizienteres RTL-Programm transformiert. Alle Übersetzer sind in C implementiert.

(a) Zeigen Sie, wie Sie diese Übersetzer auf einem x86-Rechner benutzen können, wenn Sie auf dem Rechner über einen C-Compiler verfügen.

Zeigen Sie nun, wie Sie auf einem x86-Rechner

(b) ein Pascal-Programm P in x86-Maschinensprache übersetzen können

(c) die Übersetzung von P mittels des Optimierers verbessern können

(d) ein C++-Programm Q in PPC-Maschinensprache übersetzen können.