Prof. Dr. Manfred Schmidt-Schauß
Künstliche Intelligenz/Softwaretechnologie

Fachbereich Informatik und Mathematik/ Institut für Informatik
Goethe-Universität Frankfurt am Main

Grundlagen der Programmierung 2

Sommersemester 2017

Aufgabenblatt Nr. 2

Abgabe: Mittwoch 3. Mai 2017 **vor!** der Vorlesung

Aufgabe 1 (20 Punkte)

Die Funktionen `f1` bis `f5` seien in Haskell definiert als:

```
f1 x = f2 (if x > 13 then x-180 else x+93)
f2 x = if x < 1337 then f5 x else f1 x
f3 x = if (f4 x > 20) then f5 (x+1) else f5 (x-1)
f4 x = f5 (f4 (3*x-1)) + f3 (x-1000)
f5 x = if x > 77 then f5 21 else x+42
```

a) Für welche Funktionen $\texttt{f}i, \texttt{f}j$ $(1 \leq i, j \leq 5)$ gilt:
Die Funktion $\texttt{f}i$ *referenziert* Funktion $\texttt{f}j$ *direkt*? (4 Punkte)

b) Für welche Funktionen $\texttt{f}i, \texttt{f}j$ $(1 \leq i, j \leq 5)$ gilt:
Die Funktion $\texttt{f}i$ *referenziert* die Funktion $\texttt{f}j$? (4 Punkte)

c) Welche der Funktionen $\texttt{f}i$ mit $1 \leq i \leq 5$ sind *direkt rekursiv*? (4 Punkte)

d) Welche der Funktionen $\texttt{f}i$ mit $1 \leq i \leq 5$ sind *rekursiv*? (4 Punkte)

e) Welche Paare $(\texttt{f}i, \texttt{g}j)$ mit $1 \leq i < j \leq 5$ sind *verschränkt rekursiv*? (4 Punkte)

Aufgabe 2 (20 Punkte)

a) Die Funktionen `z1`, `z2`, `z3` und `z4` seien in Haskell definiert als:

```
z1 x y z  = if y <= 33 then True
            else (z1 3 (3-y) (z*z)) - (z1 (z1 (3-x) y (9-z)) (y*34) (z/2))
z2 x y z  = if x <= y then z2 (x*2) (10+y-1) (z*3) else (y*4)
z3 x y z  = if z < 0 then (z3 2 x z) + (z3 (y+x) (13+y) (y-x)) else x-(2*z*y)
z4 x y z  = if z == 5 then (z4 (x-1) (y-1) (z-2)) + (x-2) else 1*3*4
```

Vervollständigen Sie die folgende Tabelle: (10 Punkte)

	z1	z2	z3	z4
... ist iterativ	ja / nein	ja / nein	ja / nein	ja / nein
... ist endrekursiv	ja / nein	ja / nein	ja / nein	ja / nein
... ist linear rekursiv	ja / nein	ja / nein	ja / nein	ja / nein
... ist Baum-rekursiv	ja / nein	ja / nein	ja / nein	ja / nein
... ist geschachtelt Baum-rekursiv	ja / nein	ja / nein	ja / nein	ja / nein

b) Implementieren Sie eine linear rekursive, aber nicht endrekursive, Haskell-Funktion `summe1`, welche eine Zahl x entgegennimmt und die Summe von x bis 0 zurückgibt.
(5 Punkte)

c) Implementieren Sie eine endrekursive Haskell-Funktion `summe2`, welche eine Zahl x entgegennimmt und die Summe von 0 bis x zurückgibt. Eine rekursive Hilfsfunktion könnte hierbei nützlich sein. (5 Punkte)

Aufgabe 3 (60 Punkte)

Die Funktionen `f` und `g` seien definert als

```
f x y z = if y >= 17
            then (if y < 25 then x*5 else f (x+3) (y+8) z)
            else f (x+3) (y+8) z
```

Geben Sie für alle Teilaufgaben jeweils sämtliche Reduktionsschritte sowie die jeweils verwendete Regel als Buchstabe **D**, **A** oder **I** an (**D** = Definitionseinsetzung, **A**=Arithmetische Auswertung, **I**=`if`-Auswertung).

a) Werten Sie `f 1 2 (f 13 50 42)` in *normaler Reihenfolge* aus. (15 Punkte)

b) Geben Sie die ersten 13 Schritte der Auswertung von `f 1 2 (f 13 50 42)` in *applikativer Reihenfolge* an. Wie setzt sich die Berechnung danach weiter fort? Begründen Sie Ihre Antwort, die Fortsetzung der Rechnung ist dafür nicht erforderlich. (20 Punkte)

c) Werten Sie den Ausdruck `f 1 10 (f 13 50 42)` in *verzögerter Reihenfolge* aus.
(20 Punkte)

d) Der innere `if-then-else`-Ausdruck von `f` ist redundant. Geben Sie eine abgewandelte Funktion `g` an, die komplett identische Ergebnisse liefert wie `f` – das heißt für jeweils Normalordnung, applikative Auswertung und verzögerte Auswertung soll stets das gleiche Ergebnis rauskommen – allerdings mit nur einem `if-then-else`-Ausdruck auskommt.
(5 Punkte)