ProInformatik II Funktionale Programmierung, SoSe 2016
Prof. Dr. Margarita Esponda

Endklausur

Name: …… **Vorname:**…… **Matrikel-Nr:**……

Aufgabe	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**Max. Summe**	**Note**
Max. Punkte	10	10	8	16	8	16	18	14	**100**	
Punkte										

Wichtige Hinweise:

1) Schreiben Sie in allen Funktionen die entsprechende Signatur.
2) Verwenden Sie die vorgegebenen Funktionsnamen, falls diese angegeben werden.
3) Die Klausur muss geheftet bleiben.
4) Schreiben Sie Ihre Antworten in den dafür vorgegebenen freien Platz, der unmittelbar nach der Frage steht.

Viel Erfolg!

1. Aufgabe (*10 Punkte*)

a) Erläutern Sie die Auswertungsstrategie nach Bedarf (Lazy-Evaluation)?

b) Welche wichtigen Vorteile hat eine Programmiersprache, die diese Auswertungsstrategie verwendet?

c) Was ist ein **statisches Typsystem** im Kontext von Programmiersprachen?

d) Welche Vorteile hat Haskell damit?

2. Aufgabe *(10 Punkte)*

Betrachten Sie folgende Funktionsdefinitionen:

```
span p [] = ([],[])
span p (x:xs) | p x         = (x:ys, zs)
              | otherwise = ([], (x:xs))
                              where (ys,zs) = span p xs

unfold p f g x | p x = []
               | otherwise = f x : unfold  p  f  g  (g x)
```

Was ist der **Wert** folgender Ausdrücke? Begründen Sie Ihre Antwort oder schreiben Sie mindestens drei Zwischenschritte Ihrer Berechnungen auf.

a) span (/=0) [2,3,0,1,2]

b) unfold (3<) ((==0).(`mod`2)) (1+) 1

3. Aufgabe (8 Punkte)

Definieren Sie eine Funktion **prefix** , die zwei Zeichenketten bekommt und das längste gemeinsame Prefix, falls es eines gibt, berechnet.

Anwendungsbeispiel: **prefix** "abcde" "abacde" => "ab"

4. Aufgabe (16 Punkte)

Eine Annäherung der Zahl π kann mit Hilfe folgender Reihe wie folgt berechnet werden:

$$R_n = 4 \cdot \sum_{k=0}^{n} \frac{(-1)^k}{2k+1} = 4 \cdot \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - ...\right) = \pi$$

a) Schreiben Sie eine Funktion, die mit Hilfe einer expliziten endrekursiven Hilfsfunktion die Zahl π bei Eingabe von **n** annähert.

b) Definieren Sie die Funktion nochmal unter sinnvoller Verwendung der **foldl** Funktion und Listengeneratoren.

5. Aufgabe (8 Punkte)
Betrachten Sie folgenden algebraischen Datentyp und folgende Funktionen:

```
data  Nat  =  Zero  | S Nat

add ::  Nat-> Nat-> Nat
add a Zero = a
add a (S b) = add (S a) b

npred :: Nat -> Nat
npred Zero = Zero
npred (S n) = n

foldn :: (Nat -> Nat) -> Nat -> Nat -> Nat
foldn  h  c  Zero  = c
foldn  h  c  (S n) = h (foldn  h  c  n)
```

a) Definieren Sie damit die Fibonacci-Funktion:

b) Definieren Sie eine **cutSub** Funktion (Subtraktionsfunktion mit *(a - b) = 0*, wenn *b>=a*) für natürliche Zahlen unter sinnvoller Verwendung der **foldn** Funktion.

6. Aufgabe (18 Punkte)
Betrachten Sie folgenden algebraischen Datentyp mit entsprechenden Funktionsdefinitionen:

```
data Tree a = Leaf a | Node (Tree a) (Tree a)

size :: Tree a -> Int
size (Leaf _ ) = 1                          ……..……..……..…..…… size.1
size (Node lt rt) = size rt + size lt       ……..…….. size.2

depth :: Tree a -> Int
depth (Leaf _ ) = 1                         ……..…….. depth.1
depth (Node lt rt) = max (depth lt) (depth rt) + 1 ……..… depth.2
```

Beweisen Sie mit Hilfe von struktureller Induktion, dass folgende Eigenschaft gilt:

$$\text{size t} \leq 2^{\text{depth } t}$$

7. Aufgabe *(18 Punkte)*

a) Reduzieren Sie den folgenden Lambda-Ausdruck zur Normalform:

$$(\lambda xy.\, xy(\lambda xy.\, \mathrm{y}))(\lambda xy.\, x)(\lambda xy.\, \mathrm{y})$$

b) Definieren Sie einen **CSUB** Lambda-Ausdruck, der die Subtraktion von zwei **natürlichen** Zahlen berechnet. Sie können dabei die Funktion **P** (Vorgänger) als gegeben verwenden.

c) Definieren Sie einen Lambda-Ausdruck, der die **GGT** Funktion von Euclid (größter gemeinsamer Teiler von zwei positiven natürlichen Zahlen) berechnet.

```
ggt p q | p==q = q
        | p>q  = ggt (p-q) q
        | p<q  = ggt p (q-p)
```

Sie können dabei die Funktionen **CSUB** (Subtraktion für natürliche Zahlen), **Y** (Fixpunkt-Operator) und die Vergleichsoperationen **<**, **>**, **=** als gegeben verwenden.

d) Zeigen Sie, dass folgende Lambda- und Kombinatoren-Ausdrücke äquivalent sind.

$$\lambda x.\lambda y.(xx) \quad \equiv \quad S(KK)(SII)$$

8. Aufgabe (14 Punkte)

a) Zeigen Sie, dass die Funktion **half**, die eine natürliche Zahl durch zwei teilt (ganzzahlige Division) primitiv rekursiv ist. Das bedeutet, wenn Sie für die Definition andere Hilfsfunktionen verwenden, müssen Sie auch zeigen, dass diese Hilfsfunktionen primitiv rekursiv definierbar sind.

b) Definieren Sie zusätzlich Ihre Funktionen unter Verwendung der in der Vorlesung definierten **z**, **s**, **p**, **compose** und **pr** Haskell-Funktionen.

```
type  PRFunction  =  ( [Integer] → Integer )

-- Null-Funktion
z  :: PRFunction
z xs = 0

-- Nachfolger-Funktion
s  :: PRFunction
s [x] = x+1

-- Projektions-Funktionen
p :: Integer -> [Integer] -> Integer
p 1 (a:b) = a
p n (a:b) = p (n-1) b

-- Kompositionsschema
compose :: PRFunction -> [PRFunction] -> [Integer] -> Integer
compose  f  gs  xs  =  f  [ g  xs | g <- gs ]

-- Rekursionsschema
pr :: PRFunction -> PRFunction -> PRFunction -> [Integer] -> Integer
pr  rec  g  h  (   0  :xs)  =  g  xs
pr  rec  g  h  ((n+1):xs) =  h ( (rec (n:xs)):n:xs )
```

Transformationsregeln, um Lambda-Terme in SKI-Terme zu verwandeln.

1)	T[x]	=> x	
2)	T[(E_1 E_2)]	=> (T[E_1] T[E_2])	
3)	T[λx.x]	=> I	
4)	T[λx.E]	=> (K T[E])	wenn $x \notin FV(E)$
5)	T[λx.E x]	=> (T[E])	wenn $x \notin FV(E)$
6)	T[λx.(E_1 E_2)]	=> (S T[λx.E_1] T[λx.E_2])	falls $x \in FV(E_1)$ oder $x \in FV(E_2)$
7)	T[λx.λy.E]	=> T[λx.T[λy.E]]	falls $x \in FV(E)$