# Prüfung Funktionale Programmierung

| | | |
|---|---|---|
| Datum | : | 26.07.2013, 12:30 Uhr |
| Bearbeitungszeit | : | 90 Minuten |
| Prüfer | : | Prof. Dr. Oliver Braun |
| Hilfmittel | : | Keine |
| Erreichbare Punkte | : | 90 |

Name: _____

Vorname: _____

Matrikelnummer: _____ Studiengruppe: _____

Hörsaal: _____ Platz Nr.: _____

Unterschrift: _____

Bitte kontrollieren Sie, ob Sie eine vollständige Angabe mit 5 Aufgaben auf 6 Seiten erhalten haben.

| Aufgabe | 1 | 2 | 3 | 4 | 5 | Summe |
|---|---|---|---|---|---|---|
| max. Punkte | 16 | 18 | 18 | 20 | 18 | 90 |

**Anmerkungen:**

- Sie müssen als Antworten **keine** kompletten Programme schreiben, sondern nur den explizit verlangten Teil eines Programms.
- Schreiben Sie die Lösungen in die dafür vorgesehenen Kästchen. Sollte Ihnen der Platz dabei nicht reichen, benutzen Sie die Rückseite **und vermerken Sie das im dazugehörigen Kästchen!**
- Die Rückseiten können Sie ansonsten für Nebenrechnungen etc. nutzen.

**Aufgabe 1 (16 Punkte)**

Ergänzen Sie die Typsignaturen für die folgenden Funktionen:

Hinweis: Denken Sie daran, dass in Haskell Literale überladen sein können, z.B.
`1 :: Num a => a`

(a) (4)

```
fun1 ::
fun1 x y z = z : [ a + b | a <- x, b <- y, a <= b ]
```

(b) (4)

```
fun2 ::
fun2 a b = do
    input <- readLn :: IO Double
    let a = 1.2
    print $ input + a + b
```

(c) (4)

```
import Control.Applicative ((<*>))
```

```
fun3 ::
fun3 a b = case b of
    Nothing -> a
    _       -> fmap (+) a <*> b
```

(d) (4)

```
fun4 ::
fun4 a b = filter b $ map a $ replicate 12 12
```

**Aufgabe 2 (18 Punkte)**

Definieren Sie die folgenden Funktionen:

(a) Eine Funktion die die Summe aller ungeraden Zahlen in einer Liste berechnet. Verwenden Sie mindestens eine Funktion höherer Ordnung. (5)

```
sumOnlyOdds :: [Integer] -> Integer
```

(b) Eine Funktion die das Maximum zweier Zahlen so berechnet, dass es kein Maximum gibt, wenn die Zahlen gleich sind. Verwenden Sie Guards. (5)

```
maxIfNotEqual :: Int -> Int -> Maybe Int
```

(c) Eine Funktion die drei `Maybe Int`s so addiert, dass sobald einer der Werte `Nothing` ist auch `Nothing` heraus kommt und sonst die drei `Int`s addiert und in ein `Just` verpackt werden. (8)

Tipp: `Maybe` ist Instanz der Klasse `Monad`. Nutzen Sie die `do`-Notation.

```
addThreeMaybeInts :: Maybe Int -> Maybe Int -> Maybe Int -> Maybe Int
```

**Aufgabe 3 (18 Punkte)**

Gegeben sei folgender Datentyp:

```
data Option a = Some a | None
```

(a) Geben Sie für den Datentyp `Option` eine Instanz der Klasse `Functor` an. (6)

```
class Functor f where
    fmap :: (a -> b) -> f a -> f b
```

(b) Geben Sie für den Datentyp `Option` eine Instanz der Klasse `Applicative` an. (6)
Nutzen Sie bei der Implementierung die Tatsache, dass `Option` ein `Functor` ist.

```
class (Functor f) => Applicative f where
    pure :: a -> f a
    (<*>) :: f (a -> b) -> f a -> f b
```

(c) Geben Sie für den Datentyp `Option` eine Instanz der Klasse `Monad` an. (6)

```
class Monad m where
    return :: a -> m a
    (>>=) :: m a -> (a -> m b) -> m b
    (>>) :: m a -> m b -> m b
    x >> y = x >>= \_ -> y
    fail :: String -> m a
    fail msg = error msg
```

**Aufgabe 4 (20 Punkte)**

Geben Sie das Ergebnis der folgenden Ausdrücke an. Sie können Ihre Nebenrechnungen im entsprechenden Kästchen machen. Unterstreichen Sie in dem Fall die Lösung.

(a) `foldr (+) 0 [1,0..10]` (2)

(b) `foldr1 (-) [ x+y | x <- [1..4], y <- [x,4-x] ]` (6)

(c) `take 5 $ filter (>2) $ map ((+2) . (*2)) [1..]` (6)

(d) `filter and $ map (map (flip elem [1..10])) $ replicate 10 [10..100]` (6)

**Aufgabe 5 (18 Punkte)**

Gegeben seien folgender Datentyp und folgende Typsynonyme für Filme:

```
type Title = String
data Movie = Movie
  { title :: Title
  , isRent :: Bool
  }
type Serial = Integer
type Movies = [(Serial, Movie)]
```

Gehen Sie im folgenden davon aus, dass die Seriennummer eindeutig ist und jeder Titel nur einmal in der Filmliste vorkommt.

(a) Definieren Sie eine Funktion (8)

`rentable :: Title -> Movies -> Bool`

die `True` genau dann als Ergebnis hat, wenn der übergebene Title in der Filmliste vorhanden und nicht ausgeliehen ist. Sonst ist das Ergebnis `False`.

(b) Definieren Sie eine Funktion (10)

`rent :: Title -> Movies -> (Maybe Serial, Movies)`

zum Ausleihen eines Filmes. Ist der Film nicht vorhanden oder bereits verliehen, ist die erste Komponente des Ergebnisses `Nothing`, sonst die Seriennummer. Zweite Komponente ist die (evtl. veränderte) Filmliste.