WS 2020/2021

Funktionale Programmierung

11. Übungsblatt

Prof. Dr. Margarita Esponda

---

1. Aufgabe (12 Punkte)

Definieren Sie analog zu der **parser**-Funktion für Lambda-Ausdrücke eine **ski_parser**-Funktion, sodass SKI-Ausdrucke wie folgt übersetzt werden können:

Anwendungsbeispiele:

ski_parser "x" => Var "x"

ski_parser "SKI(SKIIS)" => App (App (App S K) I) (App (App (App (App S K) I) I) S)

Zum Testen verwenden Sie die vorgegebenen Funktionen in

Ressources -> Material -> SKII_U11_Vorlage.hs.zip

Anwendungsbeispiele mit den vorgegebenen Parser/Eval-Funktionen:

skii "(S(SI(K(KI)))(K(S(KK)I)))(S(KK)I)" => "KI"

skii "(S(S(KS)(S(S(KS)(S(KK)I))(KI)))(K(K(KI))))(S(KK)I)(KI)" => "KI"

mit **skii** exp = show_expr (eval (**ski_parser** exp) False) False

2. Aufgabe (5 Punkte)

Zeigen Sie, dass die Summe der Kuben dreier aufeinanderfolgender natürlicher Zahlen durch 9 teilbar ist.

(mod (n^3 + (n+1)^3 + (n+2)^3) 9) == 0 für alle n>=0

3. Aufgabe (7 *Punkte*)

Betrachten Sie folgende Funktionsdefinitionen:

```
length :: [a] -> Int
length [] = 0                    . . . . . length.1
length (x:xs) = 1 + length xs    . . . . . length.2

replicate :: Int -> a -> [a]
replicate 0 _ = []               . . . . . replicate.1
replicate (n+1) x = x: repicate n x  . . . . . replicate.2
```

Beweisen Sie mittels vollständiger Induktion über **n** folgende Eigenschaft:

length (replicate n x) = n

4. Aufgabe (6 Punkte)

Betrachten Sie folgende Funktionsdefinitionen:

```
map f   []     = []
map f (x:xs) = (f x): map f xs

(++)  []  ys  = ys
(++) (x:xs) ys = x:(xs ++ ys)
```

Beweisen Sie mittels struktureller Induktion über **xs** folgende Eigenschaft:

map f xs ++ map f ys == map f (xs ++ ys)

Wichtiger Hinweis:

Die Lösungen sollen elektronisch (nur Whiteboard-Upload) abgegeben werden. Es ist k**eine verspätete Abgabe per Email erlaubt.**