Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion

Prof. Dr. Michael Hanus, Jan Tikovsky

Institut für Informatik

3. Übung zur Vorlesung „Programmierung“

Serie 3

Abgabe am Freitag, 15. November – 08:00

Gehen Sie bei allen Programmieraufgaben nach der Konstruktionsanleitung aus der Vorlesung vor:

1. Schreiben Sie einen kurzen Kommentar, der beschreibt was die Prozedur berechnet.
2. Geben Sie die Signaturvereinbarung der Prozedur an.
3. Definieren Sie ausreichend Testfälle (mindestens drei). Achten Sie auch darauf, dass alle Programmteile durch ihre Testfälle mindestens einmal ausgeführt werden (Keine Programmteile dunkel unterlegt nach Drücken des “Start”-Knopfes in DrRacket).

Präsenzaufgabe 1 - Fakultät

Die Funktionen `factorial`, `number-if` und `new-factorial` sind definiert durch

```
; Berechnet die Fakultät einer Zahl
(: factorial (natural -> natural))
(define factorial
  (lambda (n)
    (if (= n 0)
        1
        (* n (factorial (- n 1))))))

; Führt eine Fallunterscheidung nach einer Bedingung c durch.
; Liefert den zweiten Parameter t, falls c wahr ist und sonst den dritten e.

(: number-if (boolean number number -> number))
(define number-if
  (lambda (c t e)
    (cond (c t)
          (else e))))

;Berechnet die Fakultät einer Zahl
(: new-factorial (natural -> natural))
(define new-factorial
  (lambda (n)
    (number-if (= n 0)
               1
               (* n (new-factorial (- n 1))))))
```

Werten Sie die Terme `(factorial 1)` und `(new-factorial 1)` schrittweise entsprechend dem Substitutionsmodell aus.

Präsenzaufgabe 2 - Programmierübung

Implementieren Sie die folgenden beiden Funktionen:

1. `sum-1-to`, die die Summe der ersten `n` natürlichen Zahlen berechnet. Gibt es außer der naiven Lösung auch eine weitere, effizientere?
2. `least-common-multiple`, die das kleinste gemeinsame Vielfache zweier natürlicher Zahlen berechnet.

Aufgabe 3 - Pascalsches Dreieck

6 Punkte

Die folgende Anordnung von Zahlen ist als *Pascalsches Dreieck* bekannt:

```
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
    ...
```

Die Zahlen an den beiden Schenkeln des Dreiecks sind alle 1 und jede Zahl im Innern des Dreiecks ist die Summe der beiden schräglinks und schrägrechts darüberstehenden Zahlen. Schreiben Sie eine Funktion `pascal` mit Parametern `row` und `pos`, die die Zahl, die in der Zeile `row` und der Position `pos` im *Pascalschen Dreieck* steht, berechnet. Dabei wird immer `pos` $\leq$ `row` vorausgesetzt. Die Spitze des Dreiecks hat die Koordinaten `row` $= 0$ und `pos` $= 0$. Der Aufruf `(pascal 4 2)` soll also das Ergebnis 6 liefern. Verwenden Sie für Ihre Lösung nicht die Fakultätsfunktion!

Aufgabe 4 - Auswerten

6 Punkte

Die Funktionen `dec` und `g` sind definiert durch

```
(define dec
  (lambda (x) (- x 1)))

(define g
   (lambda (x y) (if (= x y)
                     x
                     (g x (- y 2)))))
```

Werten Sie die folgenden Terme schrittweise entsprechend dem Substitutionsmodell aus. Werten Sie dabei die Teile von Kombinationen im ersten Ausdruck von links nach rechts und die Teile von Kombinationen im zweiten Ausdruck von rechts nach links aus. Achten Sie dabei auf die korrekte Auswertung der Sonderform `if`.

1. `(g (dec (dec (+ (* 4 8) 16)))(- (/ 120 2) 12))`
2. `(dec (g (/ (- (* 7 9) 13) 2) (dec (* 13 2))))`

Aufgabe 5 - N-te Wurzel

8 Punkte

Ziel dieser Aufgabe ist es, die n-te Wurzel einer positiven reellen Zahl x mit Hilfe des Newton-Verfahrens zu approximieren. Das Newton-Verfahren definiert eine Folge $(x_i)_{i\geq 0}$ mit

$\lim_{i\to\infty} x_i = \sqrt[n]{x}$ wobei $n \in \mathbb{N}$

Die Folge ist definiert durch:

$x_0 = 1$

$$x_i = \frac{1}{n} \cdot \left(\frac{x}{x_{i-1}^{n-1}} + (n-1) \cdot x_{i-1} \right) \text{ falls } i > 0$$

Schreiben Sie eine Funktion `nth-root`, mit Parametern `n`, `x` und `eps`, die so lange Approximationsschritte durchführt, bis nach k Iterationen $\mid x_k^n - x \mid$ kleiner als die Schranke *eps* ist und dann x_k als Näherungswert für $\sqrt[n]{x}$ liefert. So soll z.B. der Aufruf `(nth-root 3 8 0.0001)` den Näherungswert für $\sqrt[3]{8}$ mit einer Genauigkeit von 0.0001 liefern.

Sie können sich bei der Implementierung am in der Vorlesung vorgestellten Newton-Verfahren zur Approximation der Quadratwurzel orientieren. Für die Berechnung der Potenzen dürfen Sie die Funktion `expnt` aus der Vorlesung für das schnelle Potenzieren einer Zahl verwenden.