Marcel Brandenburg, M. Sc.

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Klausur Einführung in die Programmierung (I167)

Quartal: (1/2020)

Name des Prüflings: **Matrikelnummer:** **Zenturie:**

Dauer: 90 Min. Seiten der *Klausur* **ohne** Deckblatt: 17 Datum: 11.03.2020

Hilfsmittel:

- KEIN Taschenrechner
- Liste der Regeln (siehe Anhang)

Bemerkungen:

- **Bitte prüfen Sie zunächst die Klausur (alle Teile) auf Vollständigkeit.**
- **Bitte lösen Sie nicht die Heftung.**

Es sind 90 Punkte erreichbar!

Aufgabe 1 (6 Punkte)

	Bewerten Sie die folgenden Aussagen:	wahr	falsch
a)	Die Pragmatik einer Programmiersprache beschäftigt sich u. a. mit der Frage, welche Ausdrucksmöglichkeiten der Sprache für die Formulierung welcher Lösungen genutzt werden sollten.		
b)	Alle zusammengesetzten Ausdrücke werden in Racket in Präfix-Notation aufgeschrieben.		
c)	Der intensive Gebrauch von Hilfsfunktionen erschwert die Pflege von Programmen.		
d)	Die Semantik funktionaler Programmiersprachen orientiert sich an der Mathematik.		
e)	Racket-Ausdrücke werden schrittweise über äquivalente Umformungen durch das Ersetzungsmodell für Funktionsanwendungen vereinfacht.		
f)	Von *Gemischten Daten* sprechen wir, wenn in Racket eine Liste Elemente unterschiedlichen Typs enthält.		

Jede richtige Antwort wird mit je 1 Punkt, jede falsche oder nicht gegebene Antwort mit 0 Punkten bewertet.

Ausdruck	Wert	Typ
`(> 4 5)`	`#false`	`Boolean`
`(lambda [x] (* x x)`	`Funktion`	`Zahl -> Zahl`

Hinweis: **Alle Ausdrücke lassen sich ohne Fehler auswerten.**

(2.1) (2 Punkte)

```
(let [(+ -)
      (/ *)]
  (+ (/ 4 5) 1))
```

Wert:

Typ:

(2.2) (3 Punkte)

```
((lambda [x y +] (+ x y)) 22 3 -)
```

Wert:

Typ:

(2.3) (4 Punkte)

```
(define y 5)
(define z -8)
(define f
  (lambda [x y]
    (let [(y 12)]
      (+ x y z) )))
(f 15 2)
```

(2.4) (5 Punkte)

```
((lambda [x]
   (lambda [y]
     (+ x y)))
 19)
```

Wert:

Typ:

Aufgabe 3 (10 Punkte)

Gegeben sei das folgende Racket-Programm:

```
(define x 2)
(define y 4)

(define z
  (lambda (x y z)
    (+ x (z y))))

(z y x (lambda (z) (+ x z)))
```

Werten Sie den Ausdruck in der letzten Zeile (durch Anwendung der formalen Auswertungsregeln) **Schritt für Schritt** aus.

Marcel Brandenburg, M. Sc.

Aufgabe 4 (13 Punkte)

Entwickeln Sie unter Anwendung aller passenden Regeln (ohne die Verwendung von Funktionen höherer Ordnung) die folgenden Funktionen.

(4.1) (5 Punkte) Eine Funktion `celsius->fahrenheit`, die die Umrechnung einer Temperatur von Grad Celsius nach Grad Fahrenheit nach folgender Formel vornimmt:

$$fahrenheit = \frac{9}{5} \cdot celsius + 32$$

(4.2) (8 Punkte) Eine Funktion, die eine Liste mit Temperaturen, gemessen in Grad Celsius, in eine Liste mit Temperaturen, gemessen in Grad Fahrenheit, umrechnet.

Aufgabe 5 (14 Punkte)

Entwickeln Sie eine Funktion `durchschnitt`, die den Durchschnitt einer Liste von Zahlen berechnet.

Anwendungsbeispiele:

```
(durchschnitt '(3 4 8)) ;=> 5
(durchschnitt '(17))    ;=> 17
```

Hinweise:

1. Funktionen höherer Ordnung dürfen **nicht** verwendet werden.
2. Für alle Funktionen dieser Aufgabe brauchen Sie nur die reine Funktionsdefinition aufzuschreiben. Sie brauchen keine Kommentare, Tests, etc. anzugeben.

(5.1) (1 Punkt) Begründen Sie, warum die Liste nicht leer sein darf.

(5.2) (3 Punkte) Formulieren Sie eine Funktionsschablone für Funktionen zur Verarbeitung von Listen von Zahlen, die nicht leer sein dürfen.

(5.3) (6 Punkte) Schreiben Sie die Funktionsdefinition für `durchschnitt` unter Verwendung von zwei geeigneten Hilfsfunktionen auf:

- Für die Berechnung der Anzahl der Elemente einer Liste benutzen Sie die Racket-Standardfunktion `length`,
- Für die Funktion zur Berechnung der Summe der Elemente verwenden Sie die Funktionsschablone aus Aufgabenteil 5.2.

(5.4) (4 Punkte) Schreiben Sie die Funktion für die Berechnung der Summe der Elemente so um, dass sie eine Hilfsfunktion mit akkumulierendem Parameter benutzt.

Aufgabe 6 (14 Punkte)

Funktionen höherer Ordnung

Gegeben sind

1. eine Strukturdefintion für Mitarbeiter*innen einer Firma

```
(define-struct mitarb [name gehalt])
```

2. eine Liste mit Mitarbeiter*innen:

```
(define firma (list (make-mitarb "Karl" 3000)
                    (make-mitarb "Rosa" 3200)
                    (make-mitarb "Klara" 3100)
                    (make-mitarb "Bertha" 4000)
                    (make-mitarb "Kurt" 3500)))
```

Hinweise:

1. In den folgenden Teilaufgaben sind **Racket-Ausdrücke** unter Verwendung von **bekannten Funktionen höherer Ordnung** zu schreiben.
2. *Racket-Ausdruck* bedeutet, dass hier keine Funktionsdefinitionen aufzuschreiben sind. Wenn z. B. ein Ausdruck für die Ermittlung des Namens

(6.1) (1 Punkt) Formulieren Sie einen Racket-Ausdruck, der die Liste `firma` in eine Liste mit den Namen verwandelt: `(list "Karl" "Rosa" "Klara" "Bertha" "Kurt")`

(6.2) (3 Punkte) Formulieren Sie einen Racket-Ausdruck, der eine Liste mit den um 10% erhöhten Gehältern der Mitarbeiter*innen in `firma` erstellt:
`(list 3300 3520 3410 4400 3850)`

(6.3) (4 Punkte) Gegeben sei die Liste der Gehälter
`(define gehaelter (list 3000 3200 3100 4000 3500))`
Formulieren Sie einen Racket-Ausdruck, der die Differenz zwischen dem höchsten und dem niedrigsten Gehalt errechnet. Das Ergebnis ist `1000`.

(6.4) (6 Punkte) Gegeben seien

- die Liste der Gehälter
  `(define gehaelter (list 3000 3200 3100 4000 3500))` sowie
- die Funktion `durchschnitt` aus Aufgabe 5

Formulieren Sie einen Racket-Ausdruck, der eine Liste mit allen Gehältern aus `gehaelter` liefert, die oberhalb des Durchschnittsgehalts liegen.

Aufgabe 7 (10 Punkte)

Gegeben sei folgende Funktion:

```
(define f
  (lambda [n]
    (cond
     [(= n 1) 1]
     [else (+ (* n n) (f (- n 1)))])))
```

Beweisen Sie mittels rekursiver Induktion, dass der Aufruf `(f n)` für jede natürliche Zahl $n > 0$ den Wert

$$f(n) = \sum_{i=1}^{n} i^2$$

liefert.

Aufgabe 8 (9 Punkte)

(8.1) (4 Punkte) Geben Sie die Wirkungsweise von `filter` an, indem Sie:

1. die Wirkungsweise durch einen deutschen Satz beschreiben,
2. einen möglichst universal verwendbaren Vertrag für diese Funktion formulieren.

(8.2) (5 Punkte) Geben Sie eine Implementierung von `filter` als rekursive Funktion in Racket an. Es genügt, die Funktionsdefinition aufzuschreiben.