Heinrich-Heine-Universität Düsseldorf
Institut für Informatik
Lehrstuhl Softwaretechnik und Programmiersprachen
Jens Bendisposto

04. Februar 2020

Klausur

Funktionale Programmierung
Wintersemester 2019/20

Nachname: _____ Vorname: _____

Matrikelnummer: _____

Zugelassene Hilfsmittel: Ein beidseitig beschriebenes A4 Blatt

Diese Klaujur enthält 10 nummerierte Seiten. Prüfen Sie bitte zuerst, ob alle Seiten vorhanden sind. Mit Ihrer Unterschrift versichern Sie, eine vollständige Klausur erhalten zu haben.

Unterschrift: _____

Schalten Sie bitte jegliche elektronischen Geräte aus

Diesen Teil bitte nicht ausfüllen:

Aufgabe	1	2	3	4	5	6	7	8	Σ
Punktzahl	10	12	8	10	10	14	6	10	80
Erreicht									

Aufgabe 1 [10 Punkte]

Gegeben sei folgende Funktion

```
(defn lol [lol]
  (for [lol lol lol lol]
    lol))
```

(a) [4 Punkte] Welche Struktur haben Eingaben, die von der Funktion fehlerfrei verarbeitet werden können?

(b) [6 Punkte] Welche Funktion oder Kombination von eingebauten Funktionen ist durch den Code implementiert?

Aufgabe 2 [12 Punkte]

In ihrem Artikel "Out of the Tar Pit" diskutieren Moseley und Marks zwei Ansätze ("Approaches to Understanding"), um Code zu verstehen.

(a) [4 Punkte] Benennen Sie die beiden Ansätze.

(b) [4 Punkte] Erklären Sie in 1-2 Sätzen den Unterschied.

(c) [4 Punkte] Erläutern Sie in Stichpunkten, welchen Einfluss Zustand auf diese beiden Ansätze hat.

Aufgabe 3 [8 Punkte]

(a) [2 Punkte] Was unterscheidet ein Wertobjekt von einem beliebigen Objekt?

(b) [6 Punkte] Warum ist es im Kontext des epochalen Zeitmodell wichtig, dass eine Identität aus einer Abfolge von Werten anstatt Objekten besteht?

Aufgabe 4 [10 Punkte]

Gegeben sei eine curried function f mit unbekannter Arität n. Schreiben Sie eine **Funktion** `decurry`, die eine solche Funktion entgegennimmt und selbst eine Funktion g zurückgibt. Die Funktion g soll eine äquivalente Funktion sein, die nicht curried ist. Werden weniger als n Argumente an g übergeben, soll der restliche Teil der curried function zurückgegeben werden. Werden zu viele Argumente übergeben, ist das Resultat unbestimmt.

Beispielaufruf:

```
user=> ((decurry (fn [a]
                   (fn [b]
                     (fn [c]
                       (+ a b c)))))
        1 2 3)
6
user=> (let [res ((decurry (fn [a]
                             (fn [b]
                               (fn [c]
                                 (+ a b c)))))
                  1)]  ;; zu wenige argumente
         ((res 2) 3))
6
```

Aufgabe 5 [10 Punkte]

Es soll eine imperative Schleife `(FORI from to anweisungen)` implementiert werden. Die Schleife hat immer eine implizite Zahlvariable i, die die Werte im Bereich von $from$ bis to (inklusive) annimmt und in den Anweisungen verwendet werden kann.

Der Aufruf soll folgendermaßen aussehen:

```
(FORI 1 10 (println i)) ;; Ausgabe der Zahlen 1 bis 10
(FORI (dec 2) (inc 9) (println i)) ;; Ausgabe der Zahlen 1 bis 10
(FORI 1 (do (println :ende) 10)
    (println i)) ;; Ausgabe von :ende gefolgt von 1 bis 10
```

Aufgabe 6 [14 Punkte]

(a) [7 Punkte] Eine Datenstruktur, um Klausurergebnisse festzuhalten, sieht wie folgt aus:

```
{"Khilipp Poerner"  {:klausur/a1 2, :klausur/a2 10, ,,,},
 "Jens Bendispasta" {:klausur/a1 7, :klausur/a2  3, ,,,},
 "Muster Loesung"   {:klausur/a1 7, :klausur/a2 11, ,,,},
 ,,,}
```

Es handelt sich um eine Map von Strings auf eine Map von mit "klausur" genamespaceten Keywords auf eine natürliche Zahl (inkl. 0).

Schreiben Sie eine Spec `::data`, die mit exakt dieser Struktur konform geht. Es wurde bereits `(require '[clojure.spec.alpha :as s])` ausgeführt.

Hinweis: `(namespace :foo/bar)` ergibt `"foo"`.

(b) [7 Punkte] Schreiben Sie eine Funktion `(avgpts data)`, die die durchschnittliche Punktzahl der Klausur berechnet. Ihre Funktion soll folgende Spec einhalten

```
(s/fdef avgpts :args (s/cat :input ::data)
               :ret number?)
```

Aufgabe 7 [6 Punkte]

Schreiben Sie einen Transducer, der die Gesamtpunkte aller Studierenden in einer Klausur ausrechnet. Die Datenstruktur in der die Ergebnisse gespeichert werden ist wieder

```
(def data {"Khilipp Poerner"  {:klausur/a1 2, :klausur/a2 10, ,,,},
           "Jens Bendispasta" {:klausur/a1 7, :klausur/a2  3, ,,,},
           "Muster Loesung"   {:klausur/a1 7, :klausur/a2 11, ,,,},
           ,,,})
```

Der Transducer soll folgendermaßen aufgerufen werden:

```
(transduce xf + data)
```

Für das Beispiel oben würde der Wert 40 als Resultat herauskommen.

Aufgabe 8 [10 Punkte]

Gegeben sei folgender Code:

```
;; Code A
(def a1 (atom nil))
(def a2 (atom nil))

(dosync
 (reset! a1 (map inc (range)))
 (reset! a2 (map inc @a1))
 (println :ende))

 ;; ----------------------------------------

;; Code B
(def r1 (ref nil))
(def r2 (ref nil))

(dosync
 (ref-set r1 (map inc (range)))
 (ref-set r2 (map inc @r1))
 (println :ende))
```

(a) [2 Punkte] Wie verhalten sich die beiden Codeblöcke A und B hinsichtlich der Terminierung? Begründen Sie Ihre Antwort in 1-2 Sätzen.

(b) [4 Punkte] Angenommen `a1` wird nebenläufig verändert. Zu welchen Zeitpunkten der Änderung gibt es eine Auswirkung auf den am Ende des `dosync` Blocks in `a2` gespeicherten Wert? Begründen Sie Ihre Antwort in 1-2 Sätzen.

(c) [4 Punkte] Angenommen `r1` wird nebenläufig verändert. Zu welchen Zeitpunkten der Änderung gibt es eine Auswirkung auf den am Ende des `dosync` Blocks in `r2` gespeicherten Wert? Begründen Sie Ihre Antwort in 1-2 Sätzen.