

Informatik 1 (Prof. Klaus Ostermann)

Gedächtnisprotokoll

12. Februar 2020

Die Bearbeitungszeit betrug 70min, keine Hilfsmittel zugelassen!

Die Bewertung erfolgt nach folgendem Schema:

Es ist immer nur eine von vier Optionen richtig, es dürfen aber mehrere Antworten angekreuzt werden. Das 1. Argument gibt die Anzahl der angekreuzten Optionen an, das 2. ob die Antwort richtig/falsch war und das 3. wie viele Punkte es dafür gibt. Es gibt negative Punkte für falsche Antworten.

(1 #true 1)	(0 #false 0)
(2 #true 0.5)	(1 #false -0.33)
(3 #true 0.21)	(2 #false -0.5)
(4 #true 0)	(3 #false -0.62)

1 Programmieraufgabe

In einer Firma gibt es Abteilungen (engl. "department") mit jeweils dazugehörigen Unter-Abteilungen (engl. "sub-department"), die wiederum Unterabteilungen besitzen können. Es sind also endlose Verschachtelungen möglich. Jede Abteilung hat Mitarbeiter (engl. "number-of-employees"). Die Summe der Mitarbeiter besteht aus den Mitarbeitern der Abteilung und der aus den Unter-Abteilungen.

1.1

Schreiben Sie eine Definition für department mit der Mitarbeiteranzahl und den Unter-Abteilungen. Es dürfen alle aus der Vorlesung bekannten Konzepte benutzt werden (insbesondere List-of). Denken Sie daran, eventuell noch weitere Daten- und Strukturdefinitionen vorzunehmen. Schreiben Sie ein Beispiel mit 2 Unterabteilungen. (3 Pkt)

1.2

Schreiben Sie eine Funktion sum-employee, die die Anzahl aller Mitarbeiter der Abteilung und zugehörigen Unterabteilungen berechnet. Schreiben Sie bei Bedarf eine Hilfsfunktion zur Berechnung der Mitarbeiteranzahl der Unterabteilungen. Schreiben Sie die Signatur auf. Tests oder eine Beschreibung sind nicht notwendig.

2 Fragen

2.1

`(cond [#true (+ 2 2)] [else 42])`

Wie lautet das Programm nach Ausführung des ersten Auswertungsschritts?

- a. `[#true (+ 2 2)]`
- b. `(cond [#true 4] [else 42])`
- c. `(+ 2 2)`
- d. `4`

2.2

`(not (and ( > 3 5 ) ( + 4 2)))`

Führt zu welchem Wert?

- a. `#true`
- b. `6`
- c. `#false`
- d. `and: question result is not true or false: 6`

2.3

Was ist Refactoring?

- a. Auswerten der Funktion mit Hilfe der Kongruenzregel
- b. Untersuchung des Programms auf die Größe der Teilmenge, die aus dem Datenuniversum genutzt wird
- c. Einbettung des Programms in eine S-Expression
- d. Zusammenfassen ähnlicher Funktionen zu einer allgemeineren Funktion

2.4

Was sind magic-numbers?

- a. Namen, denen durch Konstantendenitionen ein Wert zugewiesen wurde.
- b. Zahlenlitterale, deren Bedeutung sich nicht direkt erschließt.
- c. ...
- d. ...

2.5

Man möchte eine Ampel mit den Farben „rot“, „grün“ und „gelb“ programmieren. Welchen Datentyp nutzt man?

- a. rekursiver Datentyp
- b. Intervalltyp
- c. Produkttyp
- d. Summentyp

2.6

Wann werden laut Entwurfsrezept die Tests geschrieben?

- a. Nach den Datendefinitionen
- b. Nach der Implementierung
- c. Nach dem Funktionskopf
- d. Nach dem Template

2.7

Was ist ein Wert?

- a. `(define x 3)`
- b. `(string-append "Ich mag" "Klausuren")`
- c. `(define y (+ 3 5))`
- d. 8

2.8

Welchen Datentyp konsumiert folgende Funktion?

```
(define (f x)
  (cond
    [(number=? X) ...]
    [(string=? X "this") ...]
    [(string=? X "that") ...]))
```

- a. rekursiver Datentyp
- b. Summentyp
- c. Produkttyp
- d. Intervalltyp

2.9

Was ergibt

`(`(+ ,(+ 1 2) ,(+ 3 4)))` ?

- a. `(list '+ (list 3) (list 7))`
- b. `(list '+ (list 'unquote (list '+ 1 2)) (list 'unquote (list '+ 3 4)))`
- c. 10
- d. `(list '+ 3 7)`

2.10

Wie kann folgender Funktionsbody äquivalent ersetzt werden?

`; Boolean Boolean -> Boolean`

`(define (f b1 b2) (and b1 (or (not b2) (not #true))))`

- a. `(not b1)`
- b. `(and b1 (not b2))`
- c. `(and b1 b2)`
- d. `(or b1 b2)`

2.11

Die folgenden zwei Fragen beziehen sich auf dieses Programm:

```
(define (f x) (cond
  [(empty? x)]
  [(cons? x) (g (rest x))]))

(define (g x) (cond
  [(empty? x) empty]
  [(cons? x) (cons (first x) (f (rest x)))]))
```

Was ergibt (f (list #true 24 48 #false)) ?

- a. (list 24 #false)
- b. (list #true 24 48 #false)
- c. (list #true 48)
- d. (list (list #true 24) (list 48 #false))

2.12

Welche ist die allgemeinste Signatur für f?

- a. List-of-Number -> List-of-Number
- b. List-of-Number -> Number
- c. [X] List-of-X -> List-of-X
- d. [X] X -> X

2.13

Die folgenden 2 Fragen beziehen sich auf diese Programm:

```
(define (f x)
  (cond
    [(false? x) x]
    [else (+ x 1)]))
```

Was ist (f (f 2)) ?

- a. 2
- b. #false
- c. Typfehler
- d. 4

2.14

In der Vorlesung wurde der Datentyp MaybeNumber eingeführt. Welches ist eine mögliche Signatur für f?

- a. MaybeNumber -> MaybeNumber
- b. MaybeNumber -> Number
- c. Number -> MaybeNumber
- d. Number -> Number

2.15

Eine gültige URL wird durch diese Grammatik beschrieben:

$\langle A \rangle ::= \text{http://} \mid \text{https://} \mid \text{http://www.} \mid \text{https://www.} \mid \text{www.}$

$\langle B \rangle ::= \text{google} \mid \text{amazon} \mid \text{marketplace}$

$\langle C \rangle ::= \text{co} \mid \text{uk} \mid \text{de}$

$\langle D \rangle ::= \mid \text{home} \mid \text{uni} \mid \text{market}$

$\langle \text{URL} \rangle ::= \langle A \rangle (\langle B \rangle .)^+ \langle C \rangle^+ / \langle D \rangle$

Welches ist keine gültige URL?

- a. <http://www.google.marketplace.de/>
- b. www.google.co.uk/market
- c. <https://amazon.co.uk.de>
- d. <http://amazon.co.uk.de/market/home>

2.16

```
(define (f x y)
  (string-append x (number->string y)))
```

Was ist die allgemeine Signatur zu der Funktion?

- a. string number -> string
- b. [X Y] X Y -> X
- c. [X] X number -> X
- d. number string -> number

2.17

```
(define x (+ y 1))
```

```
(define y 3)
```

```
(define (* x y))
```

Was gibt das Programm bei Ausführung aus?

- a. 9
- b. 16
- c. 12
- d. Error: y is used before definition

2.18

Geben ist die Definition eines Auswertungskontexts für BSL mittels kontextfreier Grammatik aus dem Skript:

```
<E> ::= [ ]  
      | (<name> <v> * <E> <e>*)  
      | (cond [ <E> <e> ] {(<e> <e> }*)
```

Welche der folgenden Texte ist ein gültiger Kontext gemäß der obigen Definition?

- a. (+ (+ 7 8) 4 [] 5)
- b. (+ 4 5 [] (+ 7 8))
- c. (+ 4 (+ 7 8) [] 5)
- d. (+ [] [] []))

2.19

Was besagt die Kong-Regel?

- a. Ein Programm wird von links nach rechts und von oben nach unten ausgewertet.
- b. Falls e_1 äquivalent zu e_2 ist, dann auch $(f\ e_1)$ und $(f\ e_2)$.
- c. Man darf eine Funktion durch eine andere ersetzen, wenn sich die Bedeutung des Programms dadurch nicht verändert.
- d. Konstantendefinitionen werden direkt ausgewertet bevor sie in die Umgebung geschrieben werden.

2.20

Was ist syntaktischer Zucker?

Syntaktischer Zucker sind die Konzepte einer Programmiersprache...

- a. ...die für die Programmausführung unwesentlich sind (z.B. Kommentare). Sie müssen vor dem Parsen zu speziellen Ignore-Token entzuckert werden.
- b. ...die nicht direkt einem Prozessorbefehl entsprechen (z.B. Funktionsdefinitionen). Sie müssen vor der Ausführung zu einer Sequenz von Maschinenbefehlen entzuckert werden.
- c. ... die von einem Editor zusätzlich zum Programm angezeigt werden (z.B. farbliche Markierung nach der Syntaxprüfung). Sie müssen vor dem Ausführen wieder zum eigentlichen Programm entzuckert werden
- d. ...die leicht durch andere Konzepte ausgedrückt werden können (z.B. if durch cond). Sie müssen vor dem Ausführen in die Kernsprache entzuckert werden.

2.21

Was besagt das DRY-Prinzip?

- a. Wenn Code in einem Programm doppelt vorkommt, besteht die Gefahr, dass die Kopien nicht das gleiche berechnen wie die Originale.
- b. Programmierer sollten niemals mit Copy & Paste arbeiten. Andere
- c. Ausnutzen von copy and paste um redundante Sequenzen schneller zu programmieren
- d. Ersetzen von Mustern im Programm durch Funktionen die dieses Muster generieren

Das folgende Programm gehört zu den nächsten 4 Aufgaben:

(define-struct collision (top down left right))

; Collision is a struct: (make-collision Boolean Boolean Boolean Boolean)

2.22

Wie viele Werte hat das Datenuniversum für dieses Programm?

- a. 4
- b. 1
- c. Unendlich viele
- d. 16

2.23

Was ist eine Korrekte Form von collision?

- a. (make-collision #true 42 #false "world")
- b. (make-collision #true #false #true #true)
- c. (make-collision #false #true #false)
- d. (make-collision #true (make-collision #true #false #false #false) #true #false)

2.24

Collision ist ein ...

- a. Summentyp
- b. Produkttyp
- c. Intervalltyp
- d. Rekursiver Datentyp

2.25

Was definiert die ersten Zeilen des Programms ((define -struct collision (...)))

- a. Eine Strukturdefinition für Collision
- b. Produkttyp für Collision
- c. Keines der genannten
- d. Strukturtyp für Collision