# Gedächtnisprotokoll zur Hauptklausur Informatik 1 WiSe 19/20

## Programmieraufgabe I

Eine Abteilung besteht aus Abteilungen (engl. „departments“) und Unterabteilungen (engl. „sub-departments“). Außerdem zeichnet sie sich durch die Anzahl der Mitarbeiter aus. Eine Abteilung kann eine beliebige Anzahl an Unterabteilungen besitzen, und selbst eine Unterabteilung einer anderen Abteilung sein.
Die Gesamtzahl der Mitarbeiter einer Abteilung berechnet sich aus den Mitarbeitern der Abteilung und allen Unterabteilungen.

### Aufgabe 1

Geben Sie eine Datendefinition und eine Beschreibung einer Abteilung an, sowie ein Beispiel mit zwei Unterabteilungen.

### Aufgabe 2

Implementieren Sie eine Funktion „sum-employees“, die alle Mitarbeiter einer Abteilung berechnet, wie oben beschrieben.
Tipp: Die Aufgabe lässt sich mit Rekursion oder mit den zwei aus der Vorlesung bekannten Funktionen höherer Ordnung lösen:

```
; [X Y] (X Y -> Y) Y -> (list-of X) -> Y
(define (foldr op z) …)
; [X Y] (X -> Y) (list-of X) -> (list-of Y)
(define (map f xs) …)
```

## Programmieraufgabe II

Gegeben sei folgende Datendefinition:

```
(define-struct branch (left middle right))
(define-struct leaf (label))
```

Interpr. as Tree

### Aufgabe 1

Programmieren Sie eine Funktion (fold-tree f l t), wobei t eine Funktion ist, die auf den leaf angewandt wird, und f eine Funktion, die auf die branches angewandt wird.
Hinweis: Die Funktion muss nicht richtig implementiert sein um die nächsten zwei Aufgaben zu lösen. Sie brauchen keine Tests zu schreiben.

### Aufgabe 2

Schreiben Sie eine Funktion „product-tree“, die mit der Funktion „fold-tree“ alle Zahlen in einem Tree-of Numbers multipliziert und anschließend ausgibt. Sie brauchen keine Test zu schreiben.

### Aufgabe 3

Schreiben Sie eine Funktion, die einen Tree-of Numbers konsumiert, und den selben wieder ausgibt, es sind allerdings alle Zahlen quadriert. Benutzen Sie Ihre Funktion „fold-tree“, geben Sie keine Testfälle an.

# Multiple-Choice

(es waren immer vier Antwortmöglichkeiten gegeben, ich kann mich aber nicht immer an alle erinnern)

## Aufgabe 1

Welches Konstrukt der BSL macht es möglich mehrere Werte zu einem zusammenzufassen?

- Konstanten
- Produkttypen
- Summentypen

## Aufgabe 2

Gegeben sei folgendes Programm:

```
(define (f x)
        (lambda (y) x))
```

Zu was wertet das Programm aus wenn (map f (map f (list 1 2 3))) aufgerufen wird?

- (list 1 1 1)
- Liste mit drei Funktionen
- (list 1 2 3)
- Eine Funktion

## Aufgabe 3

Sie möchten eine Ampel mit den Ampelphasen „rot“, „gelb“ und „grün“ repräsentieren.
Welchen Datentyp wählen Sie?

- Produkttyp
- Summentyp
- Konstante

## Aufgabe 4

Was ist ein Wert?

- 3
- (+ 1 2)
- (string-append „Ich“ „liebe Klausuren“)

## Aufgabe 5

Zu was wertet `(+ 1 2 ,(+ 3 4) ,(+ 7 9)) aus?

- (list 3 7 6)
- (list ‘+ 1 2 7 6)
- (list ‘+ 1 2 (list ‘+ 3 4 (list ‘+ 7 9)))

## Aufgabe 6

Was ist eine gültige Umgebung?

- (define (f x) (+ 1 2))
  (define (y 6)
- (define (f x)  (+ 1 2))
  (define (y x))
- (define (f x) (+ 1 2))
  (define (f 6))

## Aufgabe 7

Was ist (define-struct etc (bla blu))

- Strukturdefinition
- Produkttyp
- Summentyp
- Struktur

## Aufgabe 8

Gegeben sei folgende Funktion:

```
; Number Number -> Boolean
(define (greater-than x y)
        (cond   [( > x y) #true]
                [(< x y) #false]
                [(= x y) „equals“]))
```

Was ist falsch an der Funktion? - - - -

## Aufgabe 9

Zu was reduziert das folgende Programm in einem Schritt?

```
(cond [#true (+ 2 2)]
        [(> 1 0) (+ 1 1)]]
```

- 4
- (+ 2 2)
- (cond [#true (+ 2 2)])
- (cond [#true 4]
  [(> 1 0) (+ 1 1)])

## Aufgabe 10

Für was werden Akkumulatoren verwendet?

- Um die Terminierung generativ rekursiver Rekursionen zu zeigen.
- Um Wissen, über den Kontext, indem er aufgerufen wird, zu repräsentieren.
- Um zu akkumulieren, also zu addieren.
- Um Äquivalenzen zu zeigen.

## Aufgabe 11

Gegeben sei folgendes Programm:

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

Was ergibt die Auswertung von (f (list #true 60 44 #false))?

- (list #true 60 44 #false))
- (list #true 60)
- (list #true 44)
- (list (list #true (list 60 (list 44 (list #false)))))

## Aufgabe 12

Gegeben sei folgendes Programm:

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

Zu was wertet (f (f 2)) aus?

- #false
- 2
- 4
- Fehler

## Aufgabe 13 (zu 12)

In der Vorlesung wurde der Datentyp MaybeNumber eingeführt. Was ist die allgemeinste Signatur für f?

- Number -> Number
- MaybeNumber -> Number
- MaybeNumber -> MaybeNumber
- Number -> MaybeNumber

## Aufgabe 14

Gegeben sei folgendes Programm:

```
; list-of Words word -> Boolean
; checks wether a sentence includes a forbidden word
(define (forbidden-word? sentence word) …)
; list-of Words list-of Words -> Boolean
; checks wether a sentence includes any forbidden word
(define (any-forbidden-word? sentence words)
      (cond [(forbidden-word? sentence (first words)) …]
      […]))
```

Was muss bei der zweiten Lücke eingefügt werden, damit die Funktion funktioniert?

- (any-forbidden-word? (rest sentence) words))
- (forbidden-word? sentence (rest words))
- (any-forbidden-word? sentence (rest words))
- (forbidden-word? (rest sentence) words)

## Aufgabe 14

Gegeben sei folgendes Programm:

```
(define X (+ 1 Y))
(define Y 6)
(* X Y)
```

Was ergibt die Auswertung des Programms?

- 42
- 6
- Fehler: Y ist vor seiner Definition benutzt
- 7

## Aufgabe 15
Was ist syntaktischer Zucker?

## Aufgabe 16
Was besagt die CONG-Regel?

## Aufgabe 17
Was sind magic-numbers?

## Aufgabe 18
Wann werden nach Entwurfsrezept Tests geschrieben?

## Aufgabe 19
Grammatik war gegeben:
Was ist eine gültige URL nach der Grammatik?

## Aufgabe 20
Grammatik eines Auswertungskontextes gegeben:
Was ist ein gültiger Auswertungskontext nach der Grammatik?

## Aufgabe 21
Gegeben ist folgende Funktion:

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

Was ist die allgemeinste Signatur für f?

- Number String -> String
- [X] Number X -> X
- [X Y] X Y -> X
- Number String -> Number

## Aufgabe 22
Was besagt das DRY-Prinzip?

## Aufgabe 23
Zu was wertet (or (#false (and (< 0 1) (+ 3 4)))) aus?

- #false
- 7
- #true
- Fehler: and erwartet zwei Booleans

## Aufgabe 24
Zu was kann folgendes Programm refacotort werden?

```
(define (f a b)
        (and a (or (not b) (not #true)))
```

- (not a)
- (or a b)

- (and a b)
- (and a (not b))

## Aufgabe 25

Gegeben sei folgende Funktion:

```
(define (my-animate f)
        Big bang 0
        [on-draw draw]
        [on-tick …])
```

Was für eine Funktion mit welcher Signatur muss bei … eingefügt werden?

## Aufgabe 26

Gegeben sei die aus der Vorlesung bekannte Funktion unfold:

```
; [X Y] (Y -> Bool) (Y -> Y) (Y -> X) Y -> (list-of X)
; (stop (next ... (next seed))) must yield #true after a finite number of iterations
(define (unfold stop next emit seed) …)
```

Für was ist die dritte Funktion „emit“ zuständig?

## Aufgabe 27

Gegeben sei Folgende Defintion:

```
(define-struct bla (Boolean Boolean Boolean Boolean))
```

Was ist gültig nach obiger Definition?

- (make-bla #true #false #false)
- (make-bla 34 #true 37 #false)
- (make-bla #true #false #false #true)
- (make-bla „no“)

Zudem noch 13 weitere Fragen, also 40 insgesamt.
Die Fragen entsprechen nicht exakt den originalen Fragen, sondern nur soweit ich mich erinnern kann.