EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Beispielklausur
zur Vorlesung **Informatik 1**
im Wintersemester 2015/16

Matrikelnummer:

Name:

i001

Diese Beispielklausur besteht aus einer Programmieraufgaben und 8 Fragen. Für die Bearbeitung einer Klausur diesen Umfangs hätten Sie ca. 27 Minuten Zeit.

Jede der 8 Fragen hat genau eine richtige Antwort. Für jede Frage können Sie keine Antwort, eine Antwort, oder mehrere Antworten geben indem sie die entsprechenden Felder ankreuzen. Das bedeutet:

- Wenn Sie die Antwort sicher wissen, kreuzen sie nur diese an.
- Wenn Sie sich nicht sicher sind aber einige Antworten ausschliessen können, kreuzen sie die Antworten an die korrekt sein könnten.
- Wenn Sie überhaupt nicht wissen, was die Antwort sein könnte und nichts ausschliessen können, kreuzen Sie nichts an.

Die Antworten werden so bewertet, dass zufälliges Raten null Punkte ergibt, also eine falsche Antwort Minuspunkte gibt.

Wir benutzen die Konvention dass [x] "Antwort ausgewählt" bedeutet, und sowohl [ ] als auch [■] bedeuten "Antwort *nicht* gewählt".

Programmieraufgabe I

Gegeben sind folgende Struktur- und Datendefinitionen:

```
(define-struct branch (left right))

; A Branch is a structure: (make-branch Tree Tree)
; interp. a node in the middle of a binary tree.

; A Tree is one of:
;   - Branch
;   - Number
; interp. a binary tree of numbers.
```

Aufgabe: Implementieren Sie eine Funktion, die einen `Tree`-Wert konsumiert und die Summe aller im `Tree` enthaltenen Zahlen produziert.

Verwenden Sie das Entwurfsrezept zur Funktionsdefinition!

Bewertungsschema

Diese Aufgabe zählt so viel wie 10 Fragen:

- ☐ *Signatur angegeben*
- ☐ *Signatur syntaktisch korrekt und passend zur Aufgabenstellung*
- ☐ *Aufgabenbeschreibung angegeben*
- ☐ *Aufgabenbeschreibung syntaktisch korrekt und passend zur Aufgabenstellung*
- ☐ *Funktionskopf syntaktisch korrekt und passend zur Signatur*
- ☐ *Testfälle angegeben und syntaktisch korrekt*
- ☐ *Testfälle inhaltlich sinnvoll*
- ☐ *Implementierung angegeben*
- ☐ *Implementierung benutzt sinnvolles Template*
- ☐ *Implementierung korrekt*

Tragen Sie hier nichts ein. Wir füllen das Bewertungsschema während der Korrektur aus.

Frage 1

Wieso müssen die Alternativen in Summentypen immer unterscheidbar sein?

- *a*☐ Damit man die Teilmenge des Datenuniversums bestimmen kann, die von dem Summentyp definiert wird.
- *b*☐ Damit man Programme schreiben kann, die alle Alternativen des Summentyps erzeugen.
- *c*☐ Damit man Tests schreiben kann, die alle Alternativen abdecken.
- *d*☐ Damit man Funktionen schreiben kann, die die Alternativen unterscheiden.

Frage 2

Betrachten Sie dieses Programm:

```
(define world 42)
```

Was ist der Wert des Ausdrucks `` `(('(hello ,world))) ``?

- *a*☐ `(list (list 'quote (list 'hello 'world)))`
- *b*☐ `(list (list (list 'quote (list 'hello 42))))`
- *c*☐ `(list (list (list 'quote (list 'hello 'world))))`
- *d*☐ `(list (list 'quote (list 'hello 42)))`

Programm A

Gegeben sei folgendes Programm:

```
(define (f x)
  (cond
    [(<= 3 x 5)
     (+ x 10)]
    [else
     (/ 7 x)]))
```

Die folgenden zwei Fragen beziehen sich auf dieses Programm.

Frage 3

Was ergibt die Auswertung des Ausdrucks `(f 7)` bezüglich des oben angegebenen Programmes A?

Das Ergebnis ist . . .

- *a*☐ . . . der Wert `17`
- *b*☐ . . . der Wert `10`
- *c*☐ . . . der Wert `1`
- *d*☐ . . . der Fehler „division by zero“

Frage 4

Was ergibt die Auswertung des Ausdrucks `(f 0)` bezüglich des oben angegebenen Programmes A?

Das Ergebnis ist . . .

a☐ . . . der Wert `17`
b☐ . . . der Fehler „division by zero“
c☐ . . . der Wert `1`
d☐ . . . der Wert `10`

Frage 5

Betrachten Sie folgende Strukturdefinition:

```
(define-struct foo (bar baz))
```

Welche Funktion ist für diese Struktur definiert?

a☐ `foo?`
b☐ `bar?`
c☐ `bar-foo`
d☐ `foo-bar?`

Frage 6

Was ist eine Reduktionssemantik?

Eine Reduktionssemantik beschreibt, wie man . . .

a☐ . . . eine Grammatik schrittweise in eine Datendefinition umwandeln kann.
b☐ . . . ein Programm schrittweise in einem Wert umwandeln kann.
c☐ . . . einen Wert schrittweise in eine S-Expression umwandeln kann.
d☐ . . . eine Datendefinition schrittweise in einen Produkt- oder Summentyp umwandeln kann.

Frage 7

Was ist die Signatur der folgenden Funktion?

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

Die Signatur ist . . .

a☐ `; String Number -> Number`
b☐ `; Number String -> String`
c☐ `; String String -> Number`
d☐ `; Number String -> Number`