

Informatik II, SoSe 2008

Übungsblatt 1

Abgabefrist 30.04.08 um 12:00 Uhr, Besprechung 01.05.08 - 06.05.08

Dreiwöchigen Rhythmus

Woche n :	Mi	Vorlesung zu Stoff n
		Ausgabe Übungsblatt zu Stoff $n - 1$
		Abgabetermin Übungsblatt zu Stoff $n - 2$
	Do-Di	Übungen zu Stoff $n - 2$

Sonderfälle:

- Woche 1: keine Übungsgruppen, erst mal Rechnerkennungen besorgen.
- Woche 2: Übungsgruppen im CIP, Ausgabe Übungsblatt 1
- Woche 3: Abgabe Übungsblatt 1, Besprechung Übungsblatt 1

Übungsgruppen und Teams

Jeder Teilnehmer am Übungsbetrieb ist einer Übungsgruppe fest zugeordnet. Die Übungsgruppe kann während des Semesters nicht mehr gewechselt werden.

Abgabe von Lösungen ist freiwillig. Es ist aber empfehlenswert den Stoff zu üben. Es werden Punkte vergeben, um dem Studenten ein Mass zu geben wie gut er ist. Die Punkte spielen aber keine Rolle bei der Klausurzulassung, und haben keinen Einfluss auf die Klausurnote. Die Punkte werden verteilt nach dem folgenden Schema:

nicht abgegeben	0 Pkte
abgegeben, aber Unfug	1 Pkt
akzeptabel, aber groessere Fehler	4 Pkte
gut, nur kleinere Fehler	7 Pkte
korrekt, keine Fehler	10 Pkte
besondere Leistung, ausgezeichnet	12 Pkte

Aufgabe 1-1 Terminierungsbeweis

Beweisen Sie durch vollständige Induktion über die Länge von `multiplicand`, dass der in der Vorlesung angegebene Algorithmus `integer_digit_mult` terminiert. Sie können voraussetzen, dass die verwendeten Hilfsalgorithmen für alle zulässigen Argumente terminieren.

```
function
(integer, digit) ---> integer
integer_digit_mult(multiplicand, multiplier) =
  if one_digit_integer(multiplicand)
  then (* Fall 1.1 *)
    digit_mult(multiplicand, multiplier)
  else (* Fall 1.2 *)
    (let product1 = integer_digit_mult(tail(multiplicand), multiplier)
      product2 = digit_mult(head(multiplicand), multiplier)
    in
      integer_add(product1, shift(tail(multiplicand), product2))
    )
  )
end
```

Aufgabe 1-2 Formale Spezifikation eines Algorithmus

Gegeben seien die Typen:

`digit`: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
`integer`: nicht-leere, endliche Sequenz von `digit`
`boolean`: `true` oder `false`

Gegeben seien folgende Algorithmen:

`one_digit_integer`: `integer` → `boolean`
liefert `true`, wenn das Argument eine 1-stellige `integer` ist, sonst `false`
Bsp.: `one_digit_integer(8)=true`, `one_digit_integer(13)=false`

`head`: `integer` → `digit`
liefert das linkeste `digit` eines `integer`
Bsp.: `head(238)=2`

`tail`: `integer` → `integer`
liefert das `integer`, das entsteht, wenn das linkeste `digit` des Arguments entfernt wird.
Nur sinnvoll wenn das Argument mehrstellig ist.
Bsp.: `tail(238)=38`

`append`: (`integer`, `integer`) → `integer`
liefert das `integer`, das entsteht, wenn die beiden Argumente aneinandergehängt werden
Bsp.: `append(238,4711)=2384711`

Zu spezifizieren ist folgender Algorithmus:

`shift`: (`integer`, `integer`) → `integer`
ergänzt das 2. Argument rechts um so viele Nullen, wie das 1. Argument `digit` enthält
Bsp.: `shift(4711,123)=1230000`

Der Algorithmus `shift` soll formal spezifiziert werden, wobei die gegebenen Algorithmen als Hilfsmittel verwendet werden dürfen, aber keine arithmetischen Operationen wie `*` oder `/` usw. Wenden Sie den Algorithmus auf dem Beispiel `shift(4711,123)` an.