TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Einführung in die Informatik 2
Prof. Dr. Helmut Seidl, Ralf Vogler, Stefan Schulze Frielinghaus

WS 2015/16
Übungsblatt 9

OCaml-Hausaufgaben die nicht kompilieren werden nicht gewertet! Solltet Ihr einmal in die Situtation kommen, dass sich ein Fehler in einen Teil Eurer Abgabe eingeschlichen hat, der zur Folge hat, dass sich die gesamte Datei nicht mehr kompilieren lässt, dann kommentiert diesen Teil aus und ersetzt ihn durch den Standardwert aus der Angabe (z.B. let f x = todo ()). Andernfalls wird die gesamte OCaml-Hausaufgabe mit Null Punkten gewertet.

Aufgabe 9.1 OCaml-Hausaufgabe

Die Vorlagen und Lösungen zu den Programmieraufgaben sind ab sofort über `https://github.com/vogler/info2-ha` verfügbar, damit Änderungen (sollte es während der Bearbeitungszeit welche geben) einfacher nachvollziehbar sind.

```
git clone https://github.com/vogler/info2-ha.git && cd info2-ha
git pull # update
git log # changes
```

Zur Abgabe melden Sie sich mit Ihrer TUM-Kennung auf `https://vmnipkow3.in.tum.de` an und laden ihre `ha5.ml`-Datei (nicht umbenennen!) hoch. Die Ergebnisse der Tests sind sichtbar sobald die Abgabe abgearbeitet wurde. Die endgültige Bewertung erfolgt aber erst nach der Frist.

Die Datei `batteries.ml` enthält die bisher implementierten Funktionen und ist auch in der Testumgebung verfügbar (muss nicht hochgeladen werden).

Aufgabe 9.2 OCaml-Tutoraufgaben

1. **Lambdas:** Machen Sie sich mit den folgenden Lambda-Ausdrücken vertraut und inferrieren den jeweiligen Typ. Welche Semantik haben die jeweiligen Ausdrücke?

   1.1. `fun x -> fun y -> fun z -> x (y z)`

   1.2. `fun x -> fun y -> fun z -> z (x + (fst y))`

   1.3.
   ```
   fun x -> fun y -> let rec z = fun l ->
           match l with
           | [] -> []
           | u::us -> (x u) :: (z us)
           in z y
   ```

2. **Exceptions:**

   2.1. Diskutieren Sie über die Unterschiede bei Exceptions in OCaml und Java. Was sind Vor-/Nachteile von Exceptions?

2.2. Schreiben Sie eine Funktion mit dem Typen `('a -> bool) -> 'a list -> 'a`, die das erste Element einer Liste zurück gibt, bei welchem ein Prädikat wahr ist. Das Prädikat wird als erster Parameter erwartet. Falls die Liste kein Element enthählt auf welchem das Prädikat wahr ist, soll die Exception `Not_found` geworfen werden.

2.3. Schreiben Sie eine Funktion `combine : 'a list -> 'b list -> ('a * 'b) list` die zwei Listen als Parameter nimmt und diese elementweise kombiniert. D.h. `combine [a1; ...; an] [b1; ...; bn]` liefert als Wert `[(a1,b1); ...; (an,bn)]`. Für Listen unterschiedlicher Länge soll eine Ausnahme `Invalid_argument` geworfen werden.