

Übung 12: Task List

Implementieren Sie ein Programm zur Verwaltung einer Task-Liste. Eine Task-Liste kann in Haskell als eine Liste von Paaren definiert werden:

```
[(False, "Weihnachtsgeschenke einkaufen"),  
 (False, "Vitaparcours"),  
 (True, "Burger essen im Pub")]
```

Die Task-Liste hat den Typ `[(Bool,String)]` wobei der `Bool` aussagt, ob die Aufgabe bereits erledigt ist.

```
type Task = (Bool,String)  
type TaskList = [Task]
```

Ihre Aufgabe ist es ein Programm zu schreiben um solche Listen zu verwalten. Wir leiten Sie schrittweise zur Lösung:

- a) Implementieren Sie die Funktionen `readTasks` und `writeTasks` um eine Task-Liste aus einer Datei zu lesen und darin zu speichern. Hier sind die Signaturen:

```
readTasks :: IO TaskList  
writeTasks :: TaskList -> IO ()
```

Hinweis:

Verwenden Sie die Funktionen `read` / `show` um die Daten zu konvertieren.

Falls Sie folgende Exception bekommen: "openFile: resource busy (file is locked)" lesen Sie auf der nächsten Seite weiter.

- b) Implementieren Sie die Funktion `addTask`. Sie nimmt einen `Task` und eine Liste von `Tasks` und gibt eine neue Liste von `Tasks` zurück:

```
addTask :: String -> TaskList -> TaskList
```

- c) Implementieren Sie die Funktion `numberTasks`. Sie nimmt eine `TaskList` und nummeriert sie um den `Tasks` eine eindeutige Identifikation zu geben. Die Nummerierung startet bei 0.

```
numberTasks :: TaskList -> [(Int,Task)]
```

- d) Implementieren Sie die Funktion `markDone`. Sie nimmt eine `TaskList` und einen `Int i` und gibt eine `TaskList` zurück in der der `i`'te `Task` als erledigt markiert ist:

```
markDone :: Int -> TaskList -> TaskList
```

```
markDone 1 [(False, "T1"), (False, "T2"), (True, "T3")]  
  ~> [(False, "T1"), (True, "T2"), (True, "T3")]
```

- e) Implementieren Sie das interaktive Hauptprogramm (`main`). Folgende Texteingaben sollen akzeptiert werden:

<code>list</code>	- Listet die <code>Tasks</code> nummeriert auf
<code>add Beschreibung</code>	- Fügt einen neuen <code>Task</code> hinzu
<code>done n</code>	- Markiert den <code>n</code> -ten <code>Task</code> als erledigt

*** Exception: TasksDB.txt: openFile: resource busy (file is locked)

Falls in Ihrem Code obige Exception knallt, können Sie hier die Lösung finden:
Das Problem hat mit Haskell's Lazyness zu tun. In diesem Fall bedeutet dies, dass der Inhalt des Files noch nicht fertig ausgelesen wurde aber bereits versucht wird wieder rein zu schreiben. Am einfachsten verwenden Sie folgenden Code (rot13.com):

```
ernqGnfxf = qb pbagrag <- ernqSvyr gnfxSvyr  
yrg gnfxf = vs ahyy pbagrag gura [] ryfr (ernq pbagrag)  
yratgu gnfxf `frd` erghea gnfxf
```

Das `seq` hier ist das Entscheidende. seq forciert die Evaluation des ersten Arguments und gibt als Resultat das zweite Argument zurück. Um das erste Argument zu evaluieren muss das ganze File ausgelesen und in die Liste umgewandelt werden – genau was wir erreichen wollten.