Einführung in die Informatik 2

Prof. Dr. Seidl, J. Kranz, N. Hartmann, J. Brunner — WS 2016/17

Übungsblatt 9 — Abgabefrist: 9.1.2017

Aufgabe 9.1 (P) Weihnachtliche Windungen

Implementieren Sie die Ocaml-Funktion

```
val reverse_linewise : string -> unit,
```

die einen Dateinamen als Parameter erwartet. Die Funktion soll zunächst die Datei einlesen. Anschließend werden ihre Zeilen auf Wortebene umgekehrt. Aus einer Zeile *Hallo Maus!* wird also *Maus! Hallo*. Schließlich soll der Inhalt der Datei durch die verdrehte Version ersetzt werden.

Achten Sie darauf, die Datei nach jeder Benutzung ordnungsgemäß zu schließen. Testen Sie Ihre Implementierung, indem Sie sie auf die Datei *advent.txt*, die ein besinnliches Gedicht enthält, anwenden.

Lösungsvorschlag 9.1

Die Lösung befindet sich in der Datei `ocaml/p9_sol.ml`

Aufgabe 9.2 (P) Befehle für Kamele

Ein- und Ausgabe funktioniert durch Seiteneffekte. In diesem Rahmen sind die imperativen Features von Ocaml bisweilen praktisch. In dieser Aufgabe sollen Sie daher sog. Referenzen kennenlernen, mit denen sich veränderbare Variablen repräsentieren lassen.

Besprechen Sie Referenzen kurz in der Gruppe. Implementieren Sie anschließend die Funktion

```
val decreasing : int -> int list,
```

Quellen: Camel Vector 11 von http://de.clipartlogo.com/ und http://www.zcool.com.cn/u/5/

die eine Liste absteigender Zahlen imperativ erzeugt. Sie soll dazu eine `for`-Schleife verwenden, die in jeder Iteration eine Zahl vorne an eine veränderbare Liste anhängt.

Lösungsvorschlag 9.2

Die Lösung befindet sich in der Datei `ocaml/p9_sol.ml`

Aufgabe 9.3 (P) Ausnahmezustand

In dieser Aufgabe soll ein einfaches, wenn auch etwas künstliches Beispiel für Ausnahmen betrachtet werden. Gehen Sie dabei vor wie folgt.

- Implementieren Sie die Funktion

  ```
  val apply_even_pos : (int -> int) -> int -> int,
  ```

  die eine Funktion und einen Parameter n erwartet. Sie wendet die Funktion auf den Parameter an, wenn dieser in der Definitionsmenge der Funktion liegt. Die Funktion sei auf geraden positiven Zahlen definiert. Liegt der Wert nicht in der Definitionsmenge, soll `apply_even_pos` eine Ausnahme auslösen. Dazu sollen Fehlerfälle $n < 0$, $n = 0$ und $2 \nmid n$ unterschieden werden. Definieren Sie sich die passenden Ausnahmen.

- Der Benutzer verwendet die Funktion `apply_even_pos` in der Funktion

  ```
  val apply_f_print_error : (int -> int) -> int -> int,
  ```

  die Fehler direkt auf der Konsole ausgeben soll. Dazu soll sie auf die ausgelösten Ausnahmen passend reagieren.

Lösungsvorschlag 9.3

Die Lösung befindet sich in der Datei `ocaml/p9_sol.ml`

Allgemeine Hinweise zur Hausaufgabenabgabe

Die Hausaufgabenabgabe bezüglich dieses Blattes erfolgt über das Ocaml-Abgabesystem. Sie erreichen es unter `https://vmnipkow3.in.tum.de`. Sie können hier Ihre Abgaben einstellen und erhalten Feedback, welche Tests zu welchen Aufgaben erfolgreich durchlaufen. Sie können Ihre Abgabe beliebig oft testen lassen. Bitte beachten Sie, dass Sie Ihre Abgabe stets als **eine einzige UTF8-kodierte Textdatei mit dem Namen *ha9.ml* hochladen müssen**. Die hochgeladene Datei muss ferner den Signaturen in *ha9.mli* entsprechen.

Aufgabe 9.4 (H) Caesav [8 Punkte]

In dieser Aufgabe soll eine generische Funktion zum Einlesen von CSV-Dateien implementiert werden. Eine CSV-Datei enthält eine Tabelle, wobei jede Zeile der Tabelle durch eine Zeile in der Datei repräsentiert wird. Die verschiedenen Spalten werden durch ein spezielles Zeichen getrennt; wir verwenden hier das Semikolon (;). CSV-Dateien verfügen häufig über eine Kopfzeile, die die Spaltennamen enthält; auf diese verzichten wir. Wir gehen außerdem davon aus, dass die Inhalte der Felder niemals ein Semikolon enthalten (es ist daher kein *Escaping* nötig). Eine Beispieldatei könnte wie folgt aussehen:

```
Hugo;99;2.1
Inge;11;99.1
Egon;1;0.2
```

Die Datei enthält Daten über Personen; es werden der Name, die erwartete Anzahl an Weihnachtsgeschenken und die durchschnittliche Länge der Kopfhaare in Zentimetern gespeichert. Eine CSV-Datei soll in Ocaml als Liste von Zeilen dargestellt werden. Jede Zeile ist ein *Record*, der über ein Feld pro Spalte verfügt. Sie sollen die Funktion

```
val read_csv : 'a -> 'a setter list -> string -> 'a csv
```

implementieren, die aus einem gegebenen Initialwert, einer Liste von *Setter*-Funktionen und einem Dateinamen das CSV-Objekt generiert. Die Typvariable `'a` steht hier für den Record, der die Zeilen repräsentiert. Der i-te Setter nimmt einen solchen Record und gibt einen neuen Record zurück, bei dem das i-te Feld aktualisiert wurde. Um obige Datei *xmas_hair.csv* einzulesen, könnte Ihr Programm wie folgt aussehen:

```
1 type xmas_hair_row = { name : string; expected_presents : int;
  ↪ hair_length : float }
2
3 let () =
4   let csv = read_csv {name = ""; expected_presents = 0; hair_length =
    ↪ 0.0} [
5       (fun v curr -> {curr with name = v});
6       (fun v curr -> {curr with expected_presents = int_of_string v});
7       (fun v curr -> {curr with hair_length = float_of_string v});
8     ] "xmas_hair.csv" in ...
```

Werfen Sie die Ausnahme `Column_count_does_not_match`, wenn die Anzahl der Spal-

ten in einer Zeile nicht zur Anzahl der Setter passt. Achten Sie darauf, die Datei nach Benutzung zu schließen. Fangen Sie mögliche Ausnahmen bei der Ausführung der Setter, sodass die Datei in jedem Fall geschlossen werden kann.

Lösungsvorschlag 9.4

Die Lösung befindet sich in der Datei `ocaml/ha9_sol.ml`.

Aufgabe 9.5 (H) Gruppenbildung [4 Punkte]

Implementieren Sie die Ocaml-Funktion

```
val group_by : 'a csv -> ('a row -> 'b) -> ('a row list) list,
```

die die Zeilen einer CSV-Datei (im Format aus Aufgabe *Caesav*) anhand eines Feldes gruppiert. Dazu erhält die Funktion als zweiten Parameter einen *Getter*, der aus einer Zeile dasjenige Feld, welches zur Gruppierung verwendet wird, extrahiert. Die Funktion gibt eine Liste von Gruppen zurück. Eine Gruppe enthält jeweils Zeilen, bei denen das Gruppierungselement den gleichen Wert hat. Die Reihenfolge der Zeilen bzw. Gruppen spielt keine Rolle.

Sie dürfen in dieser Aufgabe keine eigenen, rekursiven Funktionen oder Schleifen verwenden. Nutzen Sie stattdessen ausschließlich Funktionen aus dem Modul `List` für Ihre Implementierung.

Lösungsvorschlag 9.5

Die Lösung befindet sich in der Datei `ocaml/ha9_sol.ml`.

Aufgabe 9.6 (H) O Camlbaum [8 Punkte]

In dieser Aufgabe geht es darum, einen Weihnachtsbaum mittels Ocaml zu erzeugen und zu schmücken. Implementieren Sie dazu die Funktion

```
val write_xmas : string -> string -> int -> unit,
```

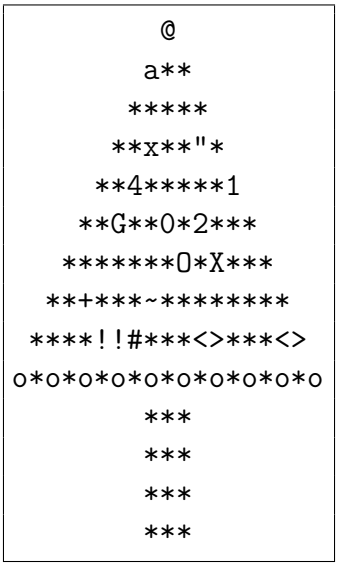
die folgende Parameter erwartet:

- Den Namen der Ausgabedatei des Christbaumes
- Den Namen einer CSV-Datei, die Schmuckstücke und ihre Positionen enthält
- Die Höhe des Baumes in Ebenen

Die Funktion soll den Weihnachtsbaum in die entsprechende Ausgabedatei schreiben. Der Baum wird mittels Buchstaben geschmückt; die CSV-Datei ist dabei im Format aus Aufgabe *Caesav*) und verwendet die folgenden drei Spalten:

Spalte:	layer	pos	decoration
Erläuterung:	Die Ebene, die geschmückt wird (von oben, ab 0)	Die Spalte des Schmuckstücks ab Baumbeginn (von links, ab 0)	Buchstabe, mit dem dekoriert wird

Nicht geschmückte Teile des Baumes sowie der Stamm (der nicht geschmückt werden kann), bestehen aus Sternchen (*). Der Baum wächst ab Spitze in jede Richtung um ein Sternchen pro Ebene; nach rechts und links soll jeweils mit Leerzeichen aufgefüllt werden. Der Stamm soll eine Höhe von $\lfloor \frac{height}{3} \rfloor + 1$ Ebenen und eine Bereite von drei Sternchen aufweisen. Ein fertig geschmückter Baum sieht z.B. aus wie folgt:



Folgende Fehler werden in dieser Aufgabe durch *Exceptions* behandelt. Werfen Sie die Ausnahme `Invalid_decoration`, wenn es sich bei einem Schmuckstück nicht um einen einzigen Buchstaben handelt. Die Ausnahme erwartet die fehlerhafte Verzierung als Parameter. Sollten nicht alle Schmuckstücke aufgehangen werden können, z.B. weil sich einige an ungültigen Positionen befinden, soll dagegen eine `Could_not_decorate`-Ausnahme erzeugt werden.

Achten Sie darauf, die Ausgabedatei nach Benutzung zu schließen. Dies muss insbesondere auch bei Fehlern gewährleistet sein.

Hinweis: Lösen Sie diese Aufgabe möglichst neben einem realen Weihnachtsbaum, den Sie als Referenz verwenden können. Öffnen Sie keine unter Ihrem Referenzbaum befindlichen Päckchen, bevor Sie die Aufgabe vollständig gelöst haben.

Lösungsvorschlag 9.6

Die Lösung befindet sich in der Datei `ocaml/ha9_sol.ml`.