

2. Übungsblatt

Ausgabe: 01.11.22**Abgabe:** 06.11.22

2.1 *Like Ice in the Sunshine.*

6 Punkte

Die Tage werden kürzer, die Blätter fallen, der Freimarkt ist da — genau die richtige Zeit ein Eiscafé zu eröffnen, findet ein Bekannter von Ihnen mit untrüglichem Geschäftssinn und einem Entrepreneur-Geist der sich auch von leider gescheiterten Geschäftsideen wie einem Brennholzverleih über das Internet (www.easyburn.com) nicht abschrecken läßt. “Aus erlesenen Zutaten komponieren wir Eiskreationen für alle Sinne” textet die Werbeabteilung (eine gemeinsame Bekannte im dritten Semester BWL). Tatsächlich klingen die Angebote lecker, und es ist ja wirklich für die Jahreszeit zu warm.

Ihr Bekannter bittet Sie, den “IT-Backbone hinter der Operation” zu entwickeln, denn er plant Großes (europaweite Expansion in 24 Monaten, er hat sich noch nie mit Kleinigkeiten aufgehalten).

Wir beginnen etwas bodenständiger und wollen den Bestellvorgang für ein Eis in Haskell modellieren. Das funktioniert wie folgt:

1. Der Bestellvorgang wird durch den Typ `Bestellung` modelliert, und beginnt mit der existenziellen Frage: in der Waffel oder im Becher?

```
waffel :: Bestellung
becher :: Bestellung
```

2. Danach werden Kugeln hinzugefügt — bis zu drei Stück in der Waffel, bis zu fünf Stück im Becher.¹

```
data Sorte = Vanille | Schoko | Erdbeer | Stracciatella | Pistazie
           deriving (Eq, Show)

kugel :: Bestellung → Sorte → Bestellung
```

Wird versucht, zu viele Kugeln hinzuzufügen, wird die Bestellung nicht verändert.

3. Alternativ kann man auch ein *Special* bestellen:

```
fuerst_P :: Bestellung
strawberry_delight :: Bestellung
choco_dream :: Bestellung
monster_mash :: Sorte → Sorte → Sorte → Sorte → Sorte → Bestellung
```

4. Am Ende kann man noch eine oder mehrere Portionen Sahne, Streusel oder Sirup hinzufügen. Dabei dürfen es höchstens zwei Portionen Sahne, und insgesamt höchstens drei Portionen Streusel und Sirup sein:

```
data Streuselsorte = Schokostreusel | Zuckerstreusel | Krokantstreusel
                  deriving (Eq, Show)

data Sirupsorte = Ahornsirup | Zimtsirup | Erdbeersirup
                deriving (Eq, Show)

sahne :: Bestellung → Bestellung
streusel :: Bestellung → Streuselsorte → Bestellung
sirup :: Bestellung → Sirupsorte → Bestellung
```

5. Auf Sahne, Streusel oder Sirup dürfen keine weiteren Kugeln hinzugefügt werden (wie sähe das denn aus?!), aber zu den Specials innerhalb der Einschränkungen von oben.

¹Eine Ausnahme ist der Special *Monster Mash*, siehe 3.

Willkommen im <i>Sunshine Ice Cream Parlour</i> !		
Unsere <i>Eissorten</i> :		Unsere <i>Tasty Toppings</i> :
• Original Bourbon Madagaskar Vanilla		• Portion Sahne €0.30
• Aulde Time Rich Flavour Chocolate		Streusel:
• Fresh and Juicy Strawberry		• Chocolate Drizzle €0.30
• Stracciatella Delizioso		• Sugar Space Glitter €0.15
• Authentic Californian Pistachio		• Krunchy Krokant €0.45
Pro Kugel €1.50		Sirup:
		• Mighty Maple €0.40
		• Sweet Cinnamon €0.20
		• Splash of Strawberry €0.25
Unsere <i>Specials</i> :		
Fürst P		€4.50
<i>Je 1 Kugel Schoko, Erdbeer, Vanille, Sahne, im Becher</i>		
Strawberry Delight		€4.90
<i>2 Kugeln Erdbeer, 1 Kugel Vanille, Erdeersirup, Sahne, im Becher</i>		
Chocolate Dream		€6.50
<i>3 Kugeln Schoko, 1 Kugel Pistazie, Ahornsirup, Krokant- und Schokostreusel, Sahne, im Becher</i>		
Monster Mash		€ 7.50
<i>5 Kugel nach Wahl, Sahne, in der Waffel</i>		

Abbildung 1: Die Karte des *Sunshine Ice Cream Parlour*

6. Am Ende bezahlen wir:

`preis :: Bestellung → Integer`

So kosten beispielsweise drei Kugeln Vanille mit Krokant und Sahne 5.25, und ein *Strawberry Delight* mit extra Sahne 5.20.

Abbildung 1 zeigt das Eisangebot und die Preise.

2.2 Business Intelligence Architecture

4 Punkte

Ihr Bekannter hatte vor einiger Zeit zwei Semester BWL studiert, und spricht daher fließend Business-Speak. Deshalb möchte er *full business intelligence capabilities* für sein *data warehouse* haben, damit er *360 degree data analysis* betreiben kann. (“Daten sind das neue Öl.”, wird er nicht müde zu betonen.) Konkret will er am Ende des Tages wissen, wieviele Kugel Eis er verkauft hat, und wieviel Umsatz er gemacht hat.²

Implementieren Sie zuerst einen linearen algebraischen Datentyp *Bestellungen* mit den Konstruktoren *Keine* und *Bestellung*, welcher die Bestellungen des Tages modelliert.

Damit implementieren Sie Funktionen, welche jeweils die Anzahl der Kugeln dieser Eissorte bzw. den Umsatz (Summe der Preise der Bestellungen) für diesen Tag zählen:

```
anzahl_kugeln_gesamt :: Bestellungen → Sorte → Integer
umsatz :: Bestellungen → Integer
```

Nehmen wie die beiden Bestellungen von oben als Tagesgeschäft, dann haben wir an dem Tag vier Kugel Vanille und zwei Kugeln Erdbeer bei einem Umsatz von 10.45 verkauft.³

Hinweise:

1. Die vorgegebenen Tests lassen sich erst übersetzen, wenn der Typ *Bestellungen* mit den Konstruktoren wie oben vorgegeben korrekt implementiert wurde.

Für die Tests müssen auch Gleichheit und eine Anzeigefunktion (*show*) auf dem Typ definiert sein; benutzen Sie dazu die magische Inkantation *deriving* (*Eq*, *Show*) wie in den anderen Typen.

2. Implementieren Sie zuerst eine Funktion

```
anzahl_kugeln :: Bestellung → Sorte → Int
```

welcher die Anzahl Kugeln dieser Sorte in *einer* Bestellung zählt. Diese kann dann auch genutzt werden, um die Gesamtanzahl Kugeln in einer Bestellung zu zählen.

Änderungen:

- *Version 1.0* Ausgegebene Version
- *Version 1.1* Zwei Fehler in den Beispielen behoben.

²Dafür interessiert sich das Finanzamt auch, aber das ist ein anderes Thema.

³Ein eher langsamer Tag, vielleicht wird es doch Herbst.