Prof. Dr. Christoph Bockisch
Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Übungen zur Vorlesung
Deklarative Programmierung: Sommersemester 2018

Nr. 4, Abgabe bis 21.5.2018 11:59 Uhr

Hinweis: Ab diesem Übungszettel müssen stets Datendefinitionen und Beispiele zu den verwendeten Typen sowie Signatur, Aufgabenbeschreibung und Tests zu den implementierten Funktionen angegeben werden. Halten Sie sich nicht daran, wird pro Aufgabe pauschal 1 Punkt abgezogen (Pro Aufgabe werden aber nicht weniger als 0 Punkte vergeben).

Aufgabe 4.1: Arbeiten mit Strukturen | 4 Punkte

In der Vorlesung haben Sie die posn-Struktur kennengelernt. Wir möchten nun einige Funktionen auf dieser Struktur implementieren:

a) `(vec-add v1 v2)`, die die übergebenen Vektoren addiert.

b) `(vec-sub v1 v2)`, die die übergebenen Vektoren subtrahiert.

c) `(vec-skal-mult s v)`, die den übergebenen Vektor mit dem Skalar s multipliziert.

d) `(vec-norm v)`, die den übergebenen Vektor normiert.

Aufgabe 4.2: Die Liga der außergewöhnlichen Summentypen | 4 Punkte

In dieser Aufgabe sollen Sie **schrittweise** die fünf Schritte des Entwurfsrezepts für Summentypen durchführen. Achten Sie darauf, dass Sie jeden einzelnen Schritt benennen und dokumentieren. Geben Sie einen geeigneten Summentypen an, der die folgenden Bedingungen erfüllt:

Ein Auto verbraucht bei einer Geschwindigkeit von unter durchschnittlich 120km/h 8 Liter auf 100km. Befindet sich die Geschwindigkeit zwischen 120km/h und 160km/h, so liegt der Verbrauch bei 10 Liter auf 100km. Ist die Geschwindigkeit allerdings höher als 160km/h, so steigt der Verbrauch linear an (0.55 Liter pro 10km/h).

Aufgabe 4.3: Guardians of the Reduction | 4 Punkte

Betrachten Sie das folgende Programm. Reduzieren Sie den Ausdruck `(reveal THEDOCTOR)` in der letzten Zeile zu einem Wert und geben Sie alle Reduktionsschritte an. Geben Sie zu jedem Schritt jeweils an, welche Regel Sie angewendet haben, sowie das Ergebnis der Reduktion. Programmteile, die durch den Reduktionsschritt nicht verändert werden und die für den folgenden Schritt nicht relevant sind, können Sie durch ... abkürzen.

```
(define-struct hero (supername realname race))
(define (reveal hero) (cond
    [(string=? (hero-race hero) "Martian") (string-append
    (hero-supername hero) " is " (hero-realname hero)
    " from Mars.")]
    [(string=? (hero-race hero) "Kryptonian") (string-append
    (hero-supername hero) " is " (hero-realname hero)
    " from Krypton.")]
    [(string=? (hero-race hero) "New God") (string-append
    (hero-supername hero) " is " (hero-realname hero)
    " from New Genesis.")]
    [(string=? (hero-race hero) "Timelord") (string-append
    (hero-supername hero) " is " (hero-realname hero)
    " from Gallifrey.")]
    [(string=? (hero-race hero) "Sayajin") (string-append
    (hero-supername hero) " is " (hero-realname hero)
    " from Vegeta.")]
    [(string=? (hero-race hero) "Old One") (string-append
    (hero-supername hero) " is " (hero-realname hero)
    " from R'lyeh.")]
    [true (string-append (hero-supername hero) " is "
    (hero-realname hero) " from Earth.")]))

(define THEDOCTOR (make-hero "The Doctor" "Unkown" "Timelord"))
(reveal THEDOCTOR)
```

Aufgabe 4.4: Rocket Game | 4 Punkte

Wir wollen die interaktive Rakete vom letzten Zettel nun zu einem kleinen Spiel weiterentwickeln:
Die Rakete startet in der Mitte der Szene und fliegt auf den oberen Rand zu. Bei einem Klick in die Szene soll die Rakete ihre Richtung ändern und auf die Position des Klicks zufliegen. Das Spiel endet, wenn die Rakete den Rand der Szene berührt. Um das ganze etwas schwieriger zu gestalten, soll die Geschwindigkeit der Rakete mit jedem Klick um 1 zunehmen.
Das letzte Feature, das wir implementieren möchten ist ein Punktezähler, der mit jedem Tick um 1 erhöht wird.

a) Im ersten Schritt müssen wir alle benötigten Informationen (Positionsvektor, Richtungsvektor, Punkte, Geschwindigkeit) in einer geeigneten Struktur ablegen ($\rightarrow$ WorldState). Definieren Sie eine solche Struktur!

b) Implementieren/modifizieren Sie die Funktionen `render` und `end-of-world`, sodass sie mit der neuen Struktur arbeiten. Ist das Spiel beendet, sollen der Text „GAME OVER“ und die erreichten Punkte angezeigt werden (Sie können sich auch einen kreativen „Game Over“-Screen überlegen, der diese Kriterien erfüllt).

c) Jeder Tick soll den Punktestand um 1 erhöhen und die Rakete um `Geschwindigkeit` Pixel in Richtung des Richtungsvektors bewegen. Implementieren Sie die entsprechende on-tick-`event` Funktion!

d) Implementieren Sie nun die Klick-Handler-Funktion, die Geschwindigkeit und Richtungsvektor gemäß der Beschreibung modifiziert. Achten Sie hierbei darauf, dass der Richtungsvektor stets normiert ist.

Hinweis: Greifen Sie bei der Implementierung auf die Funktionen aus Aufgabe 4.1 zurück. Sollten Sie Aufgabe 3.4 nicht bearbeitet haben, finden Sie eine Musterlösung im ILIAS.