Informatik I: Scheme-Programmieraufgaben zum Selbststudium

2. Datentypen und Records
(Bei Fragen können Sie sich an die Tutoren des betreuten Programmierens wenden)

Aufgabe 1: Geometrische Körper

a) Definieren Sie Record-Prozeduren für die zusammengesetzten Datentypen `wuerfel`, `kugel`, `zylinder` sowie `kegel`. Dem Würfel- bzw. Kugelkonstruktor soll jeweils die Seitenlänge bzw. der Radius mitgegeben werden, den Zylinder- und Kegelkonstrukturen der Radius der Grundfläche sowie die Höhe. Halten Sie sich dabei an die aus Vorlesung und BPs bekannten Namenskonventionen und schreiben Sie Signaturen für alle Konstruktoren.

   Definieren Sie den gemischten Datentyp `koerper`, der alle oben genannten Datentypen einschließt.

b) Definieren Sie eine Prozedur `(: koerper-volumen (koerper -> number))`, die für jeden beliebigen Körper das Volumen berechnet. Die dazu benötigten Formeln können Sie einer Formelsammlung (oder Wikipedia) entnehmen. Benutzen Sie die Annäherung $\pi \approx 3.14159$.

   Zur Kontrolle können Sie folgende Testfälle verwenden:

```
(check-expect (koerper-volumen (make-wuerfel 2)) 8)
(check-within (koerper-volumen (make-kugel 2)) 33.510 1e-3)
(check-within (koerper-volumen (make-zylinder 2 4)) 50.266 1e-3)
(check-within (koerper-volumen (make-kegel 2 4)) 16.755 1e-3)
```

c) Definieren Sie eine Prozedur `(: koerper-oberflaeche (koerper -> number))`, die für jeden beliebigen Koerper die Oberfläche berechnet. Die dazu benötigten Formeln können Sie einer Formelsammlung (oder Wikipedia) entnehmen. Benutzen Sie die Annäherung $\pi \approx 3.14159$.

   Zur Kontrolle können Sie folgende Testfälle verwenden:

```
(check-expect (koerper-oberflaeche (make-wuerfel 2)) 24)
(check-within (koerper-oberflaeche (make-kugel 2)) 50.266 1e-3)
(check-within (koerper-oberflaeche (make-zylinder 2 4)) 75.398 1e-3)
(check-within (koerper-oberflaeche (make-kegel 2 4)) 40.666 1e-3)
```

Aufgabe 2: Punkte und Geraden im kartesischen Raum

a) Definieren Sie Record-Prozeduren für den zusammengesetzten Datentyp `vec2d`, der im Folgenden zur Definition von Vektoren und Punkten benutzt werden soll. Die beiden Konstruktorargumente sollen die horizontale und vertikale Vektorkomponente darstellen. Halten Sie sich an die aus Vorlesung und BPs bekannten Namenskonventionen und

schreiben Sie Signaturen für alle Konstruktoren.
Hinweis: Sie sollten für diese Aufgabe ganz von der Verwendung von Fließkommazahlen absehen, da diese mit einer gewissen Ungenauigkeit behaftet sind, was zu unerwünschten Ergebnissen und Fehlern führen könnte.

Definieren Sie sich den Nullvektor `(: vec2d0 vec2d)`.

b) Definieren Sie die Prozedur `(: vec2d=? (vec2d vec2d -> boolean))`, die zwei Vektoren auf Gleichheit testet.

Zur Kontrolle können Sie folgende Testfälle verwenden:

```
(check-expect (vec2d=? (make-vec2d 0 0) vec2d0) #t)
(check-expect (vec2d=? (make-vec2d 3 4) (make-vec2d 4 3)) #f)
```

c) Definieren Sie die Prozeduren `(: vec2d+ (vec2d vec2d -> vec2d))` zum Addieren und `(: vec2d- (vec2d vec2d -> vec2d))` zum Subtrahieren zweier Vektoren. Definieren Sie außerdem die Skalarmultiplikation `(: vec2d* (number vec2d -> vec2d))`.

Zur Kontrolle können Sie folgende Testfälle verwenden:

```
(check-expect (vec2d+ (make-vec2d 3 5) (make-vec2d 2 -1)) (make-vec2d 5 4))
(check-expect (vec2d+ vec2d0 (make-vec2d 2 -1)) (make-vec2d 2 -1))
(check-expect (vec2d- (make-vec2d 3 5) (make-vec2d 2 -1)) (make-vec2d 1 6))
(check-expect (vec2d- vec2d0 (make-vec2d 2 -1)) (make-vec2d -2 1))
(check-expect (vec2d* 2 (make-vec2d 2 -1)) (make-vec2d 4 -2))
(check-expect (vec2d* 0 (make-vec2d 2 -1)) vec2d0)
```

d) Definieren Sie Record-Prozeduren für den zusammengesetzten Datentyp `gerade`. Halten Sie sich dabei an die aus Vorlesung und BPs bekannten Namenskonventionen und schreiben Sie Signaturen für alle Konstruktoren. `(make-gerade s r)` soll dabei eine Gerade mit dem Stützvektor `s` und dem Richtungsvektor `r` (beide vom Typ `vec2d`) erzeugen. Fassen Sie die Typen `vec2d` (der in diesem Sinne als Punkt aufgefasst wird) und `gerade` im gemischten Datentyp `obj2d` zusammen.

e) Definieren Sie die Prozedur
`(: schnitt-typ (obj2d obj2d -> (one-of "punkt" "gerade" "leer")))`, die für zwei beliebige Objekte der Sorte `obj2d` entscheidet, ob der Schnitt eine Gerade, ein Punkt oder leer ist. Es ist dabei sehr hilfreich, zunächst einige Hilfsprozeduren für die einzelnen Konstellationen der Eingabetypen zu schreiben. Zusätzlicher Hinweis: Mit der Bestimmung linearer (Un)abhängigkeit können Sie viel anfangen!

Überlegen Sie sich Testfälle, die möglichst alle Fälle abdecken. Nicht getestete Codesequenzen sind in DrRacket blau hervorgehoben.