Lehrstuhl für Kognitive Modellierung
Prof. Dr. Martin V. Butz
Dr. Jan Kneissler

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Informatik II

Programmierübungsblatt 2

Ausgabe: Mo, 05.05.2014 Abgabe: 19te Kalenderwoche

Für die Bearbeitung der Übungsaufgaben haben Sie 90 Minuten Zeit. Bitte geben Sie Ihre Lösungen der Aufgaben rechtzeitig im Vorlesungssystem unter dem Punkt Präsenzübungen ab. Dazu müssen Sie sich mit Ihrem Benutzernamen und Passwort am Vorlesungssystem anmelden. Sie können mehrmals innerhalb der Bearbeitungszeit Ihre Lösungen abgeben, nur die letzte Version Ihrer Abgabe wird gespeichert. Wenn Sie mit den Übungen fertig sind, bleiben Sie bitte an ihrem Platz sitzen und warten, bis ein Tutor ihre Lösungen bewertet. **Bei Fragen zur Rechnerbenutzung oder zu den Übungsaufgaben wenden Sie sich bitte jederzeit an einen der TutorInnen!**

Aufgaben (insgesamt 24 Punkte)

Notiz: Halten Sie sich beim Programmieren in Scheme an die eingeführte Konstruktionsanleitung:
a) Kurzbeschreibung als Kommentar (mittels: "; ..."");
b) Signatur (der "Vertrag" mittels: "(: ...)");
c) Testfälle (mittels "(check-within ...)" oder "(check-expect ...)" oder ...)
d) Prozedur Gerüst und Rumpf
Benutzen Sie Hilfsprozeduren, wenn Teilprobleme gelöst werden müssen!

Notiz: Die aktuelle `Info2_SchemeEssentials_XX.rkt` Datei steht Ihnen als weitere Hilfestellung zur Verfügung.

2.1 Kartenstich (14 Punkte)

In einem hypothetischen Kartenspiel gelten die folgenden Regeln. Eine Spielkarte besteht aus einem *Bild* und einer *Farbe*. Die möglichen Bilder sind *Ass, König, Dame, Bube, Zehn, Neun, Acht, Sieben, Sechs*, wobei die Bilder nach absteigender Wertigkeit geordnet sind. Das *Ass* hat also die höchste Wertigkeit, die *Sechs* die niedrigste. Die möglichen Farben sind *Kreuz, Pik, Herz* und *Karo*.
Beim Spiel ist immer eine Farbe festgelegt, die die *Trumpffarbe* ist. Eine Spielrunde läuft dann wie folgt ab: Der Spieler A, der an der Reihe ist, spielt eine Karte mit einer bestimmten Farbe aus. Danach spielen auch die weiteren Spieler $(B, \ldots)$ jeweils eine Karte aus. Wer den Spielzug gewinnt (d.h. *sticht*), entscheidet sich nach den folgenden Kriterien:

- Wenn die weiteren Spieler Karten mit der gleichen Farbe wie A ausspielen, wird nach der Wertigkeit der Karten entschieden: die Karte mit der höchsten Wertigkeit gewinnt.
- Wenn die Karte von A nicht die Trumpffarbe hat und einer der weiteren Spieler $B, \ldots$ eine Karte der Trumpffarbe ausspielt, hat A direkt verloren.
- Wenn ein Spieler $B, \ldots$ eine Karte mit einer Farbe ausspielt, die nicht der Farbe von A entspricht und nicht die Trumpffarbe ist, hat dieser Spieler automatisch verloren.

Dabe können Sie annehmen, dass immer unterschiedliche Karten gespielt werden.

Schreiben Sie Prozeduren um zu entscheiden, ob A einen Stich macht. Dazu sind die folgenden Teilprobleme zu lösen:

a) (4 Punkte) Schreiben Sie eine Prozedur `sticht-bild?`, die zwei Bilder übergeben bekommt und entscheidet, ob das erste Bild eine höhere Wertigkeit hat als das zweite. `sticht-bild?` liefert `#t` genau dann, wenn das erste Bild eine höhere Wertigkeit hat als das zweite.

Wählen Sie dazu zunächst eine geeignete Repräsentation für die möglichen Bildwerte und definieren Sie Namen für die einzelnen Bilder.

Stellen Sie mit Hilfe der Signatur sicher, dass nur gültige Werte für Bilder übergeben werden.

Ein korrekter Test wäre zum Beispiel: `(check-expect (sticht-bild? ass sieben) #t)`

b) (6 Punkte) Schreiben Sie eine Prozedur `sticht-2?`, die für zwei Spieler A und B genau dann `#t` zurückliefert, wenn A den Stich macht. Die Prozedur bekommt als Parameter die Farben und Bilder der Karten von A und B sowie die aktuelle Trumpffarbe.

Stellen Sie mit Hilfe der Signatur sicher, dass nur gültige Werte für Bilder und Farben übergeben werden.

Test: `(check-expect (sticht-2? koenig pik neun karo karo) #f)`

c) (4 Punkte) Schreiben Sie eine Prozedur `sticht-3?`, die für drei Spieler A, B und C genau dann `#t` zurückliefert, wenn A den Stich macht. Die Prozedur bekommt als Parameter die Farben und Bilder der Karten von A, B und C sowie die aktuelle Trumpffarbe.

Stellen Sie mit Hilfe der Signatur sicher, dass nur gültige Werte für Bilder und Farben übergeben werden.

Test: `(check-expect (sticht-3? koenig pik ass kreuz neun karo herz) #t)`

Befolgen Sie die **Konstruktionsanleitung** für Prozeduren und Fallunterscheidungen!

Achten Sie bei den Testfällen auf eine **vollständige Überdeckung** (siehe blau markierte Bereiche nachdem Start ausgeführt wurde)!

Hinweis: Denken Sie daran, dass Sie Signaturen selbst erzeugen können! Die *Spezialform* `(signature <sig>)` liefert die Signatur mit der Notation `<sig>`. Eine so erzeugte Signatur kann auch mittels `define` an einen Namen gebunden werden.
Beispiel: `(define foo (signature (one-of 1 2)))`

Abgabe: Programm: `cards.rkt`

2.2 Note Berechnen (10 Punkte)

Damit Sie am Ende des Semesters Ihren Notendurchschnitt berechnen können, bietet es sich an, diese Berechnung mit Hilfe einer Racket-Prozedur durchzuführen.
Programmieren Sie eine Prozedur `notendurchschnitt`, die aus den drei Parametern Präsenzübungsleistung (%), Übungsleistung (%) und Klausurnote (Note) den Notendurchschnitt berechnet.
Für die Berechnung des Notendurchschnitts gilt:

- Die Präsenzübungsleistung (in Prozent) geht mit 25% in die Berechnung des Durchschnitts ein.
- Die Übungsleistung (in Prozent) geht mit 25% in die Berechnung des Durchschnitts ein.
- Die Klausurleistung (in Prozent) geht mit 50% in die Berechnung des Durchschnitts ein. Die resultierende gewichtete Leistung in Prozent wird dann mit Hilfe der Umrechnung in Tabelle 1 in eine Note umgerechnet. Eine Klausur mit weniger als 40% der Punkte gilt als nicht bestanden und resultiert automatisch in einem Durchschnitt von 5.0.

`Prozentsatz` (x)	`Note`
$90 \leq x$	1.0
$85 \leq x < 90$	1.3
$80 \leq x < 85$	1.7
$75 \leq x < 80$	2.0
$70 \leq x < 75$	2.3
$66 \leq x < 70$	2.7
$62 \leq x < 66$	3.0
$58 \leq x < 62$	3.3
$54 \leq x < 58$	3.7
$50 \leq x < 54$	4.0
$x < 50$	5.0

Tabelle 1: Umrechnung von Prozentsätzen in Noten (wird für die Info II Endnote nicht unbedingt so berechnet werden).

Ein beispielhafter Test wäre : `(check-within (notendurchschnitt 70 100 67) 2 0.001)`
Weitere checks könnten z.B. testen, ob die Prozentzahlen 70 100 100 eine 1 ergeben, 70 0 50 eine 5 ergeben oder 100 100 39 eine 5 ergeben.
Befolgen Sie die **Konstruktionsanleitung** für Prozeduren und Fallunterscheidungen und **abstrahieren Sie Teilprobleme** (z.B. die Umrechnung von Prozent in Noten)!

Achten Sie bei den Testfällen auf eine **vollständige Abdeckung** aller möglichen Fälle!

Abgabe: Programm `noten02.rkt`