IX. Übungsblatt "Informatik I" WS 95/96

Christian Schulte, Prof. Gert Smolka

Ausgabe: 15. Dezember 1995

Abgabe: 22. Dezember 1995 vor der Vorlesung (Programme per E-Mail)

Aufgabe IX.1 (Erzeugen von Zählern (3)) In der Vorlesung haben Sie bereits einen *Zähler* kennengelernt: Ein Zähler ist eine Prozedur die wiederholt ohne Argumente aufgerufen wird. Bei jedem Aufruf gibt der Zähler eine Zahl zurück, die um eins größer ist als der vorher zurückgelieferte Wert. Schreiben Sie eine Prozedur `(erz-zaehler anfang)`, die einen Zähler erzeugt, der bei `anfang` beginnt zu zählen. Erzeugt man einen Zähler mit `(define z (erz-zaehler 4))`, so soll `(z)` zuerst zu `4` evaluieren, dann zu `5` und so weiter.

Aufgabe IX.2 (Total Recall (4))

1. Schreiben Sie eine Prozedur `(vorher x)`, die sich bei Applikation das Argument `x` merkt und zu dem vorherigen Argument evaluiert. Beim ersten Mal soll das Symbol `*erstes-mal*` zurückgeliefert werden. Zum Beispiel soll `(vorher 'a)` zu `*erstes-mal*` evaluieren, dann `(vorher 'b)` zu `a`, dann `(vorher 1)` zu `b`.

2. Schreiben Sie eine Prozedur `(erz-vorher start)` mit dem eine Prozedur `vorher` wie oben erzeugt werden kann. Mit `(define vorher (erz-vorher '*erstes-mal*))` können wir zum Beispiel eine Prozedur `vorher` wie oben erzeugen.

Aufgabe IX.3 (Flips und Flaps (6))

1. Schreiben Sie eine Prozedur `flip` (ohne Argumente), die bei der ersten Applikation `1`, bei der zweiten `0`, bei der dritten wieder `1`, bei der vierten wieder `0`, und so weiter, zurückliefert.

2. Schreiben Sie eine Prozedur `erz-flip`, mit der die obige Prozedur `flip` erzeugt werden kann. Das heißt, wenn wir `(define flip (erz-flip))` ausführen, soll sich `flip` wie oben verhalten.

3. Nehmen Sie an, daß der Interpreter die folgenden Definitionen ausgeführt hat:

   ```
   (define flip (erz-flip))
   (define flap (flip))
   (define (flop) (flip))
   (define flup flip)
   (define (flep) flip)
   ```

   Zu welchem Wert evaluiert jeder der folgenden Ausdrücke (wenn sie von links nach rechts evaluiert werden):

   `flap`, `flop`, `flup`, `flep`, `(flap)`, `(flop)`, `(flup)`, `(flep)`, `flap`, `(flup)`, `(flop)`.

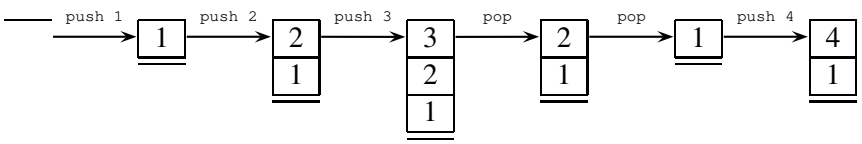
Aufgabe IX.4 (Zählerobjekt (4)) In der Vorlesung haben Sie ein Objekt zur Verwaltung eines Bankkontos kennengelernt. Hier sollen Sie ein Zählerobjekt programmieren, das als enkapsulierten Zustand eine Zahl enthält, die durch Methoden des Objektes gesetzt, inkrementiert, dekrementiert oder abgefragt werden kann.
Definieren Sie eine Prozedur `(erz-zaehler anfang)`, die ein Zählerobjekt erzeugt, das mit der Zahl `anfang` als Zustand initialisiert wird. Ein Zählerobjekt `z` soll die folgenden Methoden besitzen:

- `((z 'setze) 4)` setzt den Zustand auf `4`.
- `((z 'inc))` erhöht den Zustand um eins.
- `((z 'dec))` verringert den Zustand um eins.
- `((z 'hole))` evaluiert zu dem aktuellen Zustand des Objektes.

Aufgabe IX.5 (Stack (4)) Ein *Stack* ist eine Datenstruktur, die die folgenden Operationen erlaubt:

- Testen, ob der Stack leer ist (`empty?`).
- Ablegen eines Elements oben auf dem Stack (`push`).
- Zugriff auf das oberste Element des Stacks (`top`).
- Entfernen des obersten Elements des Stacks (`pop`).

Graphisch kann man sich einen Stack und die Operationen auf diesem so vorstellen:



Implementieren Sie eine Prozedur `(erz-stack)`, die ein Stackobjekt erzeugt, das initial als Zustand den leeren Stack hat. Das Stackobjekt soll dabei die oben angegebenen Methoden haben.

Aufgabe IX.6 (Umdrehen von Listen mit einem Stapel (4)) In der vorherigen Aufgabe haben Sie Stacks kennengelernt. Stacks funktionieren nach dem sogenannten LIFO-Prinzip (**l**ast **i**n - **f**irst **o**ut): Das Element was man zuletzt auf dem Stapel mit `push` abgelegt hat, kann man zuerst wieder mit `pop` entfernen.
Man kann zum Umdrehen von Listen einen Stack benutzen, in dem man zuerst alle Elemente der Liste mittels `push` in dem Stapel ablegt, und dann mittels `top` und `pop` alle Elemente vom Stack entfernt und daraus die Liste in umgekehrter Reihenfolge aufbaut. Realisieren Sie diese Idee in einer Prozedur `(reverse xs)` die zu einer Liste evaluiert, die die Elemente der Liste `xs` in umgkehrter Reihenfolge enthält.

Die Übung der Gruppe 2 (Georg Demme) findet außergewöhnlich am Montag, dem 18. Dezember, in Raum +1.01, Geb. 43.1 (DFKI) statt.
Alle anderen Übungen finden in der kommenden Woche wie gewohnt statt.