Laborator 3

Variabile in Scheme. Formurile let, let*, letrec, named let

Documentatie:

Help>Help Desk>Manuals>Teach Yourself Scheme in Fixnum Days>capitolele 5, 6.

Rezolvarea problemelor:

1. Comparatie let, let*, letrec.

R:

Pentru cele 3 formuri consideram formatul: (nume (initializari) (corp))

a) comparatie let / let*.

- pentru let: variabilele definite in (initializari) sunt vizibile doar in (corp)
- pentru let*: variabilele definite in (initializari) sunt vizibile in (corp). De asemenea in (initializari) sunt vizibile toate variabilele definite in (initializari) inainte de variabila curenta

b) letrec:

- „letrec” se foloseste pentru functii locale care se apeleaza reciproc si pentru functii locale recursive
- pentru functii locale nerecursive si care nu apeleaza alte functii locale se pot folosi „let”, „let*” sau „letrec”

2. Scrieti o functie care sa afiseze toate numerele naturale intre n1 si n2 inclusiv, (n1 ≤ n2) ordonate crescator. Se cer doua variante: una cu letrec, alta cu named let.

R:

```
(define numara1
  (lambda (n1 n2)
    (letrec
      ((aux (lambda (i)
              (if (> i n2)
                  '()
                  (cons i (aux (+ i 1)))))))
      (aux n1))))
```

```
(define numara2
  (lambda (n1 n2)
    (let aux ((i n1))
         (if (> i n2)
            '()
            (cons i (aux (+ i 1)))))))
```

3. Rescrieti functia list-position din capitolul 6.3, folosind letrec in loc de named let.

R:

```
(define list-position
  (lambda (o l)
    (letrec ((loop (lambda (i l)
      (if (null? l) #f
         (if (eqv? (car l) o) i
            (loop (+ i 1) (cdr l)))))) (loop 0 l))))
```

4. Scrieti o functie care reintoarce pozitiile tuturor aparitiilor unui element intr-o lista. Rezultatul va fi o lista.

R:

```
(define list-positions
  (lambda (o l)
    (letrec ((loop
            (lambda (i l1)
              (if (null? l1) '()
                 (if (eqv? (car l1) o)
                    (cons i (loop (+ i 1) (cdr l1)))
                    (loop (+ i 1) (cdr l1)))
                 ))))
      (loop 0 l))))
```

5. Scrieti o functie care face reuniunea a doua multimi. Multimile sunt date ca liste.

R:

```
(define reun
  (lambda (l1 l2)
    (if (null? l1)
        l2
        (if (list-position (car l1) l2)
            (reun (cdr l1) l2)
            (cons (car l1) (reun (cdr l1) l2))))))
```

6. Scrieti o functie care face intersectia a doua multimi. Multimile sunt date ca liste.

R:

```
(define inters
  (lambda (l1 l2)
    (if (null? l1)
        '()
        (if (list-position (car l1) l2)
            (cons (car l1) (inters (cdr l1) l2))
            (inters (cdr l1) l2)))))
```