Упражнение 1 по Лисп (newlisp) – 29.09.2004 г.

1. Оценяване на изрази – константи, символи (атоми), примитиви (вградени функции), списъци, ламбда изрази, специални форми. Прилагане на функция. Свързване на символ със стойност (set, define); ' (quote).

```
> 12
12
> -3.4
-3.4
> 567e-3
0.567
> "proben string"
"proben string"
> "\072\069\076\076\079"
"HELLO"
> nil
nil
> true
true
> +
+ <4075B0>
> (lambda (x) (+ x 10))
(lambda (x) (+ x 10))
```

```
> 'abc
abc
> '"abc
"abc
> (mul 5 5 4)
100
> (1 2 3)
Invalid function :  (1 2 3)
> ((lambda (x) (add x 2)) 7)
9
> (set 'f (lambda (x y) (add x y)))
(lambda (x y) (add x y))
> (f 2 3)
5
> a
nil
> (set 'a 5)
5
```

```
> a
5
> (define b (+ 3 5))
8
> b
8
> (set 'a 'b)
b
> a
b
> b
8
> (set a 0)
0
> a
b
> b
0
```

Основно правило за оценяване на списък: 1. Оценява се първият елемент на списъка и тази оценка трябва да е примитив или ламбда-израз (функция). 2. Оценяват се следващите елементи на списъка (от втория до последния). 3. Прилага се функцията (получена от 1) към оценките от 2.

2. Дефиниране на функция – define. Примитиви, реализиращи аритметични и логически операции и основни функции: +, -, *, /, %, add, sub, mul, div, abs, log, exp, sqrt, pow, sin, cos; <, >, >=, <=, =, !=, and, or, not.

Специални форми за реализиране на разклонения – if и cond.

Задачи:

1.1. Дефинирайте функция за пресмятане на: $f(x)=\begin{cases} x^2-x+4, & x>2 \\ \dfrac{1}{x}, & 1\le x\le 2 \\ \ln|x|+2.x+7, & x<1, x\neq 0 \end{cases}$.

```
(define (f x)
   (cond ((> x 2) (add (sub (mul x x) x) 4))
         ((>= x 1) (div 1 x))
         ((!= x 0) (add (log (abs x)) (mul 2 x) 7))
   ))
```

1.2. Дефинирайте функция, която реализира следната разклонена функция:

$$f(x)=\begin{cases} -1, & x>20 \text{ или } x<-20 \\ x-4, & -20\le x\le 10 \\ \dfrac{10}{x^2+5}, & 10<x\le 20 \end{cases}$$

```
(define (f x)
  (cond ((or (> x 20) (< x -20)) -1)
        ((<= x 10) (sub x 4))
        ((div 10 (add (mul x x) 5)))
))
```

1.3. Дефинирайте функция, която реализира следната функция:

$$f(x)=\begin{cases} x^{10}-x^5+5, & x>5 \\ 100-x^{20}-x^{15}, & x\le 5 \end{cases}$$

```
(define (f x)
  (if (> x 5) (add (sub (pow x 10 ) (pow x 5)) 5)
              (sub 100 (pow x 20) (pow x 15)) ))
```

1.4. Дефинирайте функция, която реализира следната рекурсивна функция:

$$s(w)=\begin{cases} -w, & w\le -20 \\ 2.w, & -20<w<3 \\ (w-1)\cdot s(w-1), & w\ge 3 \end{cases}$$

```
(define (s w)
  (cond ((<= w -20) (sub w))
        ((< w 3) (mul 2 w))
        ((mul (sub w 1) (s (sub w 1))))
  ))
```

3. Локални и глобални символи (!особености на версията).

Символите формални параметри на функция са локални. При прилагане на функция стойността им се запазва и след извършване на изчислението се възстановява. (Пример 1.) Останалите символи в тялото на функцията са глобални (Примери 2 и 3.) Локални променливи, освен в списъка с параметри на функцията могат да бъдет зададени и чрез спец. форма let.

Пример 1.

```
> (set 'x 1)
1
> (define (h x) (set 'x (+ x 10)))
(lambda (x) (set 'x (+ x 10)))
> x
1
> (h 5)
15
> x
1
```

Пример 2.

```
> (set 'b 0)
0
> (define (h x) (set 'b x))
(lambda (x) (set 'b x))
> b
0
> (h 6)
6
> b
6
```

Пример 3.

```
> (define (f x) (mul x 10))
(lambda (x) (mul x 10))
> (define (g x) (define (f x) (add x 5)) (mul x (f x)))
(lambda (x)
  (define (f x) (add x 5))
  (mul x (f x)))
> f
(lambda (x) (mul x 10))
> (g 2)
14
> f
(lambda (x) (add x 5))
```

В следващите две дефиниции pr1, h и z са локални (проверете!).

```
(define (pr x y)
  (let ((pr1)(h 10))
    (define (proc1 z) (* z x h))
    (+ (proc1 x) (proc1 y))))
```

```
(define (pr x y pr1 h)
   (set 'h 10)
   (define (proc1 z) (* z x h))
      (+ (proc1 x) (proc1 y)))
> (pr 1 2)
30
```

Рекурсивен (отложени изчисления до връщане на резултат от рекурсивното прилагане на функцията) **и итеративен** (няма отложени изчисления, а на всяка стъпка се получава частичен резултат, който се доближава до търсения) **процес на изчисление**. И в двата случая функциите са рекурсивни.

1.Използване на рекурсията за осъществяване на итерация – слага се допълнителен параметър на функцията, който при всяко извикване актуализира стойността си и осъщесвява итерацията.

2.1. Дефинирайте функция, която по зададено n пресмята n-тия член на рекурсивната редица: $a_1=6$; $a_2=-4$; $a_n=3.a_{n-1}-2.a_{n-2}.n.(n-1)$, $n>2$.

Рекурсивно решение:

```
(define (red n)
  (cond ((= n 1) 6)
        ((= n 2) -4)
        ((> n 2) (sub (mul 3 (red (- n 1))) (mul 2 (red (- n 2)) n (- n 1))))
  ))
```

Итеративно решение:

```
(define (red1 n iter)
  (define (iter k b c)
    (if (> k n) c
        (iter (+ k 1) c (sub (mul 3 c) (mul 2 b k (- k 1))))
    ))
  (if (= n 1) 6 (iter 3 6 -4))
)
```

2.2. Дефинирайте функция за пресмятане на n-ти елемент на редицата на Фибоначи.

Итеративно решение:

```
(define (fibon n iter)
  (define (iter k x y)
    (if (= k n) y (iter (+ k 1) y (+ x y))))
  (iter 0 0 1))
```

2.3. Дефинирайте функция, която намира най-малкото число от редицата на Фибоначи, което е по-голямо от зададено n.

```
(define (fibN n iter)
  (define (iter x y)
    (if (> y n) y (iter y (+ x y))))
  (iter 1 1))
```

2.4. Дефинирайте функция, която по зададени реално x и цяло n намира стойността за x на n-ия полином от редицата:

$P_0(x)=1$, $P_1(x)=x$, $P_n(x)=((2.n-1).x.P_{n-1}(x)-(n-1).P_{n-2}(x))/n$, $n>1$.

Рекурсивно решение:

```
(define (p x n)
  (cond ((= n 0) 1)
        ((= n 1) x)
        ((> n 1) (div (sub (mul (- (* 2 n) 1) x (p x (- n 1)))
                           (mul (- n 1) (p x (- n 2))))
                      n))
  ))
```

Итеративно решение:

```
(define (p1 x n cycle)
   (define (cycle k y z)
      (if (> k n) z
          (cycle (+ k 1) z
             (div (sub (mul (- (* 2 k) 1) x z)
             (mul (- k 1) y)) k))
      ))
    (if (> n 0) (cycle 2 1 x) 1)
)
```

2.5. Дефинирайте функция с параметри n и x, която генерира линейно итеративен процес за намиране на стойността на полинома:
$P(x,n) = n.x^n+(n-1).x^{n-1}+ \ldots +2.x^2+x.$

2.6. Дефинирайте функция, която намира десетичното число, което се получава при записване на десетичните цифри на естествено число n в обратен ред (водещите нули се игнорират).

2.7. Дефинирайте функция, която при дадено n проверява дали в редицата числа $i^4+3.i.n^2+n^4$, i=1,…,n има такова, което се дели на x.

2. Използване на специални форми за цикъл – for, dotimes, while, until, dolist за осъществяване на итерация.

2.8. Дефинирайте функция, която по зададено n>0 пресмята сумата:

$$S_n(x)=\sum_{k=0}^{n}\frac{(-1)^k.x^{2k+1}}{(2k+1)!}=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots+\frac{(-1)^n.x^{2n+1}}{(2n+1)!}\ .$$

Първо решение (итеративен процес):
```
 (define (sum x n iter)
  (define (iter i y s)
   (if (> i (+ n n 1)) s
    (iter (+ i 2) (mul -1 x x (div y (+ i 1) (+ i 2))) (add s y))))
  (iter 1 x 0))
```
Второ решение (с конструкция for):
```
(define (sum1 x n)
 (if (= n 0) x
     (let ((y x)(s x))
          (for (k 1 n 1)
               (set 'y (div (mul -1 y x x) (mul (add k k) (add k k 1))))
               (set 's (add s y))
          )
     )))
```
2.9. Да се дефинира функция за пресмятане на безкрайната сума

$$x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\ldots=\sum_{n=0}^{\infty}\frac{(-1)^n.x^{2n+1}}{(2n+1)!}$$ с точност ε. Забележка: $$\sum_{n=0}^{\infty}\frac{(-1)^n.x^{2n+1}}{(2n+1)!}=\sin(x)\ .$$

Първо решение (итерат. процес):)
```
(define (mysin x eps iter)
  (define (iter i y s)
     (if (< (abs y)eps) s
         (iter (+ i 2) (div (mul y x x -1) (mul (+ i 2) (+ i 1))) (add s y))))
  (iter 1 x 0))
```
Второ решение (с while (until)):
```
(define (sinW x eps)
  (let ( (y x) (S x) (j 2) )
       (while (>= (abs y) eps)
                (set 'y (div (mul -1 x x y) j (add j 1)))
                (set 'j (add j 2))
                (set 'S (add S y))
       )
  ))
```
Вместо (while (>= (abs y) eps) ...)
може да се използва (until (< (abs y) eps) ...)

2.10. Да се дефинира функция $f(x,n)=\prod_{j=0}^{n-1}(x+j)$.

Решение с dotimes:
```
(define (multx x n)
  (let ( (M 1) )
       (dotimes (i n) (set 'M (mul M (add x i)))) ))
```

Упражнения 3.2 и 4 по Лисп (newlisp) – 11,18.10.2004 г.

Задачи за обработка на списъци

3.1. Конструирайте изразите по-долу като използвате функцията **cons**, необходимите символи и константи:
а) ((a) 1 2 (e r))
```
 > (cons (cons 'a) (cons 1 (cons 2 (cons (cons 'e 'r)))))
```
б) (1 2 (3 4) ((d f)))
в) (a (b (c) 1) ((2)))
г) (1 2 (b (c) 1) (3))

3.2. Конструирайте примерите от зад. 1 като използвате функцията **list**.
```
 > (list (list 'a) 1 2 (list 'e 'r))
```

3.3. Напишете израз, който конструира списъка ((a) 2 ((1 2) b 1)). Можете да използвате само cons (или list), празен списък, костанти и символи.
```
 > (cons (cons 'a)(cons 2 (cons (cons (cons 1 2)(cons 'b 1)))))
 > (list (list 'a) 2 (list (list 1 2) 'b 1))
```

3.4. Нека символа f е свързан със стойността (1 a (b (h))(d)), т.е. имаме (set ′f ′(1 a (b (h))(d))). Като използвате **first** и **rest** (и f), напишете израз, чиято оценка е символа h.
```
 > (first (first (rest (first (rest (rest f))))))
```

3.5. Като използвате **first**, **rest** и израза ((a) 2 ((1 3) b 1)), напишете израз, чиято оценка е 3.
```
 > (set 'x '((a) 2 ((1 3) b 1)))
 > (first (rest (first (first (rest (rest x))))))
```

3.6. Дефинирайте функция, преброяваща колко елемента на даден списък са цели числа.
```
(define (intnum x)
  (if (empty? x) 0
      (+ (intnum (rest x)) (if (integer? (first x)) 1 0))
  ))
```
С използване на спец. форма за цикъл **dolist**.
```
(define (nums L)
  (let ( (S 0) )
       (dolist (elm L)
               (set 'S (+ S (if (integer? elm) 1 0)))
       )
  ))
```

3.7. Дефинирайте функция, която преброява елементите на списък x, които са списъци от n елемента.
```
(define (br x n)
  (cond ((empty? x) 0)
        ((and (list? (first x))(= (length (first x)) n) (+ (br (rest x) n) 1))
        ((br (rest x) n))
 ))
```

3.8. Дефинирайте функция, която намира сумата (произведението) на елементите на списък, които са цели числа (които са числа по-малки от 5 или по-големи от 20).

3.9. Дефинирайте функция, която намира сумата на целите числа в списък, съдържащи се и в елементите-списъци (т.е. целите числа на всички нива).
```
(define (sumel1 l)
  (cond ((= l '()) 0)
        ((integer? (first l)) (+ (first l) (sumel1 (rest l))))
        ((list? (first l)) (+ (sumel1 (first l)) (sumel1 (rest l))))
        ((sumel1 (rest l)))
  ))
```

3.10. Напишете функция, която пресмята сумата от четните числа в списък.

3.11. Напишете функция, която пресмята $\prod_{k=1}^{n}(a_k - k)$ за даден списък $(a_1\, a_2 \ldots a_n)$.

3.12. Дефинирайте функция, която проверява дали в един списък има 2 съседни равни елемента. (Да се контролира дали се подава списък!)

```
(define (neib2eq x)
  (if (list? x)
      (cond ((= (rest x) '()) nil)            //или ((< (length x) 2) nil)
            ((= (first x) (first (rest x))))
            ((neib2eq (rest x)))) ))
```

3.13. Дефинирайте функция, която проверява дали един списък е монотонно намаляващ.

```
(define (namal? x)
   (if (<= (length x) 1) true
       (and (>= (first x) (first (rest x))) (namal? (rest x)))) ))
```

3.14. Дефинирайте функция, която конструира списък от n еднакви елемента x.

```
(define (dupln x n)
  (cond ((= n 0) '())
        ((> n 0) (cons x (dupln x (- n 1))))))
```

3.15. Дефинирайте функция, която конструира списък от числата от n до 1.

3.16. Дефинирайте функция, която конструира списък от числата от n до 1, всяко записано по два пъти един след друг: напр. (5 5 4 4 3 3 2 2 1 1).

```
(define (make2 n)
  (cond
   ((= n 0) '())
   ((> n 0) (cons n (cons n (make2 (- n 1)))))))
```

3.17. Дефинирайте функция, която конструира списък от числата от 1 до n

```
(define (make3 n pom)
  (define (pom i)
    (cond
      ((> i n) '())
      ((<= i n) (cons i (pom (+ i 1))))))
  (pom 1))
```

3.18. Напишете функция, която по зададено число n конструира списък на числата от 1 до n, записани по следния начин: (2 1 4 3 …).

3.19. Дефинирайте функция, която от дадени три списъка (a_1 a_2 a_3 …), (b_1 b_2 b_3 …) и (c_1 c_2 c_3 …) конструира четвърти по следния начин: ((a_1 b_1 c_1)(a_2 b_2 c_2)(a_3 b_3 c_3)…). (При изчерпване на единия списък елементите, останали в другите се игнорират.)

```
(define (makel x y z)
  (if (or (= x '())(= y '())(= z '())) '()
      (cons (list (first x)(first y)(first z))(makel (rest x)(rest y)(rest z))) ))
```

3.20. Дефинирайте функция, която от дадени два списъка (a_1 a_2 a_3 …) и (b_1 b_2 b_3 …) конструира трети по следния начин: (a_1 b_2 a_3 b_4 …).

```
(define (slivane1 x y)
   (cond
      ((= x '()) '())
      ((< (length y) 2) (cons (first x)))
      ((= (length x) 1) (cons (first x) (cons (first (rest y)))))
      ((cons (first x)
             (cons (first (rest y))
                   (slivane1 (rest (rest x)) (rest (rest y))))))
   ))
```

II начин

```
(define (sliv x y)
  (cond ((= x '()) '())
        ((= y '()) (cons (first x)))
        ((cons (first x)(sliv (rest y) (rest x))))
  ))
```

3.21. Дефинирайте функция, която дублира един след друг елементите на списък, които са символи: (1 a c 3 (a d))-->(1 a a c c (a d)).

```
(define (duplsym x)
    (cond ((= x '()) '())
          ((symbol? (first x)) (cons (first x)
                                     (cons (first x) (duplsym (rest x)))))
          ((cons (first x) (duplsym (rest x))))
    ))
```

3.22. Напишете функция, която конструира нов списък от даден чрез замяна на всяко срещане на x в списъка с y.

3.23. Дефинирайте функция, която конструира нов списък състоящ се от елементите на даден, записани в обратен ред.

3.24. Дефинирайте функция, която намира максималният елемент на числов списък.

3.25. Дефинирайте функция, която увеличава с 2 всеки 3-ти елемент на числов списък.

Упражнение 5 по Лисп (newlisp) – 25.10.2004 г.

Използване и дефиниране на функции от по-висок ред

I. Използване на filter, map, apply, eval .

5.1. Дефинирайте функция, която намира произведението на елементите на числов списък.

```
I н. (define (multlist x) (eval (cons 'mul x)))
II н.(define (multlist1 x) (apply mul x))
```

5.2. Дефинирайте функция, която намира минималния елемент на числов списък.

5.3. Дефинирайте функция (като използвате filter), която намира броя на нечетните елементи на целочислен списък.

```
I н.
(define (odd? x) (!= (% x 2) 0))
(define (br_odd x) (length (filter odd? x)))
II н.
(define (numodd x)
  (length (filter (lambda (x) (!= (% x 2) 0)) x)))
```

5.4. Дефинирайте функция (като използвате filter), която намира сумата от елементите на даден списък, които са цели числа.

5.5. Дефинирайте функция (като използвате filter), която конструира списък от онези елементи на даден списък, които са числа на Фибоначи.

5.6. Дефинирайте функция, която от три списъка с равна дължина $(a_1\ a_2\ \ldots\ a_n)$, $(b_1\ b_2\ \ldots\ b_n)$ и $(c_1\ c_2\ \ldots\ c_n)$ конструира списъка $((a_1\ b_1\ c_1)\ (a_2\ b_2\ c_2)\ \ldots\ (a_n\ b_n\ c_n))$.

```
(define (sl x y z) (map list x y z))
```

5.7. Да се дефинира функция, която увеличава с 2 всички елементи на числов списък, които са по-големи от 50 и по-малки от 100.

5.8. Дефинирайте функции, които от списък от вида $(a_1\ a_2\ \ldots\ a_{n-1}\ a_n)$ получават списък:

а) $((a_1\ a_1)\ (a_2\ a_2)\ \ldots\ (a_n\ a_n))$

б) $((a_1\ a_2)\ (a_2\ a_3)\ \ldots\ (a_{n-1}\ a_n))$

5.9. Дефинирайте функция, която от списък с елементи непразни списъци от числа получава нов списък, в който всички числа са удвоени.

```
(define (f Lst)
  (map (lambda (x) (map (lambda (y) (mul 2 y)) x)) Lst))
```

5.10. Даден е списък от списъци от числа. Дефинирайте функция, която:

а) създава списък от сумите на елементите на всеки от подсписъците на дадения;

б) създава списък от минималните елементи на всеки от подсписъците на дадения;

в) намира максималния от минималните елементи на подсписъците.

II. Дефиниране на функции от по-висок ред

5.11. Дефинирайте функция от по-висок ред, реализираща итерацията за пресмятане на суми от вида на зад.8, упр. 3.1 и я използвайте за пресмятане на сумата:

$$S_n(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots + \frac{x^n}{n!}.$$

```
(define (iter i n x y s nexti nexty)
   (if (> i n) s
       (iter (nexti i) n x (nexty y i x) (add s y) nexti nexty)))

Използване (за пресмятане на посочената примерна сума):
(define (inc1 i) (+ i 1))
(define (ny y i x) (div (mul y x) (inc1 i)))

(define (suma x n)(iter 0 n x 1 0 inc1 ny))
```

5.12. Дефинирайте функция от по-висок ред, която намира сумата на елементите на даден числов списък, които отговарят на дадено условие и с нейна помощ намерете сумата на елементите на списък, които са цели числа, по-големи от 2.

```
(define (sumels l us)
  (cond
   ((= l '()) 0)
   ((us (first l)) (+ (first l) (sumels (rest l) us)))
   ((sumels (rest l) us))))

(define (prov x)
  (and (integer? x)(> x 2)))
(define (sum l)
  (sumels l prov))
```

5.13. Дефинирайте функция, която конструира списък, съдържащ елементите на четна позиция от даден списък (първият ел. е на 1-ва позиция), които отговарят на дадено условие.

```
(define (filterE usl x)
  (cond ((empty? (rest x)) '())//у-вието ще бъде true, ако x е празен или с 1 ел-т
        ((usl (first (rest x)))
         (cons (first (rest x)) (filterE usl (rest (rest x)))))
        (true (filterE usl (rest (rest x)))))
)
```

5.14. Дефинирайте функция, която от един списък конструира друг, състоящ се само от онези елементи на първия, които отговарят на списък от условия.

```
(define (nfilter usllst lst)
  (if (= usllst '()) lst
      (nfilter (rest usllst) (filter (first usllst) lst))))
```

5.15. (Табулиране на функция в даден интервал.) Дефинирайте функция, която конструира списък от двойки (x,f(x)) за x=a,a+h,a+2h,…,b и я приложете за табулиране на функцията $f(x)=\begin{cases}\ln x, & x>0\\ 8, & x=0\\ x^2+1{,}5, & x<0\end{cases}$ в интервала [1,5] със стъпка 0,5.

```
(define (tab func a b h)
  (if (> a b) '()
      (cons (list a (func a))(tab func (add a h) b h))
  ))

(define (f x)
  (cond ((> x 0) (log x))
        ((= x 0) 8)
        ((add (mul x x) 1.5))
  ))
(tab f 1 5 0.5)
```

5.16. Дефинирайте функция, която от две дадени ламбда-функции получава ламбда-функцията суперпозиция на дадените, т.е. от f(y) и g(z) генерира h(x)=g(f(x)). Прилагането на функцията за f(y), дефинирана чрез ламбда-израза (lambda (y) (mul y y)) и g(z), дефинирана чрез (lambda (z) (add z 10)) дава в резултат суперпозицията им - ламбда-израза

```
(lambda (x) ((lambda (z) (add z 10)) ((lambda (y) (mul y y)) x)))
```

I н.

```
(define (NL1 Ly Lz)
  (let ((f)) (eval (list 'define '(f x) (list Lz (list Ly 'x))  ))
  ))
```

II н.

```
 (define (NL2 Ly Lz)
  (append (lambda (x)) (list (append (list Lz) (list (list Ly 'x))))))
```