Lisp 3-1

Funkce pro práci se seznamy - zkrácený zápis sekvence car a cdr:

(cdr '(a (b (c d)) e)) -> ((b (c d)) e)
(car '((b (c d)) e)) -> (b (c d))
(cdr '(b (c d))) -> ((c d))
(car '((c d))) -> (c d)
(cdr '(c d)) -> (d)
(car '(d)) -> d

(car (cdr (car (cdr (car (cdr '(a (b (c d)) e))))))) -> d
(cadr (cadadr '(a (b (c d)) e))) -> d
(cadar (cdadr '(a (b (c d)) e))) -> d
(cadadr (cadr '(a (b (c d)) e))) -> d

Predikáty:

eq,neq - identita v paměti + malá čísla
eql,neql - stejný objekt nebo stejná čísla
equal - srovnání struktury, listy pomocí eql
plusp - test na kladnost operandu
listp - test zda je operand seznam

(eq 1 1) -> t
(eq 1000 1000) -> t
(eq 1000000 1000000) -> t
(eq 1000000000 1000000000) -> nil
(eql 1000000000 1000000000) -> t
(eq '(a b) '(a b)) -> nil
(eql '(a b) '(a b)) -> nil
(equal '(a b) '(a b)) -> t

Proměnné:
(setq jméno_proměnné výraz) - první argument se nevyhodnocuje, druhý ano
(set jméno_proměnné výraz) - vyhodnocují se oba

(setq a '(1 2 3))
a -> (1 2 3)
(setq b '(5 6))
b -> (5 6)
(setq a b)
a -> (5 6)
b -> (5 6)

(setq x 'a)
x -> a
(set x '(7 8))
x -> a
a -> (7 8)

Lisp 3-2

Výběr prvku z dané pozice v seznamu:

```
(defun at_index(n s)
  (cond ((zerop n) (car s))
        ((null s) nil)
        (t (at_index (- n 1) (cdr s)))
) )
```

příklad: (at_index 0 '(a b c d)) -> a
příklad: (at_index 2 '(a b c d)) -> c
příklad: (at_index 4 '(a b c d)) -> nil

Seznam bez posledního prvku:

```
(defun bez_posledniho(s)
  (cond ((null s) nil)
        ((null (cdr s)) nil)
        (t (cons (car s) (bez_posledniho (cdr s))))
) )
```

příklad: (bez_posledniho '(a b c d)) -> (a b c)

Binomické koeficienty B(n,k) = B(n-1,k-1) + B(n-1,k):

```
(defun binom(n k)
  (cond ((zerop n) 1)
        ((zerop k) 1)
        ((equal n k) 1)
        (t (+ (binom (- n 1) (- k 1)) (binom (- n 1) k)))
) )
```

příklad: (binom 5 3) -> 10

Vložení prvku do seznamu na danou pozici:

```
(defun insert(a n s)
  (cond ((zerop n) (cons a s))
        ((null s) nil)
        (t (cons (car s) (insert a (- n 1) (cdr s))))
) )
```

příklad: (insert 'e 0 '(a b c d)) -> (e a b c d)
příklad: (insert 'e 2 '(a b c d)) -> (a b e c d)
příklad: (insert 'e 4 '(a b c d)) -> (a b c d e)

Test na přítomnost v seznamu:

```
(defun je_prvkem(x s)
  (cond ((null s) nil)
        ((equal x (car s)) t)
        (t (je_prvkem x (cdr s)))
) )
```

příklad: (je_prvkem 'a '(b c a d)) -> t
příklad: (je_prvkem 'a '(b c d)) -> nil

Výběr kladných čísel ze seznamu:

```
(defun kladna(s)
  (cond ((null s) nil)
        ((plusp (car s)) (cons (car s) (kladna (cdr s))))
        (t (kladna (cdr s)))
) )
```

příklad: (kladna '(-3 1 2 -6 4)) -> (1 2 4)

Lisp 3-3

Test na lichá/sudá čísla:

```
(defun liche(n)
  (cond ((zerop n) nil)
        ((equal n 1) t)
        (t (liche (- n 2)))
) )

(defun sude(n)
  (cond ((zerop n) t)
        ((equal n 1) nil)
        (t (sude (- n 2)))
) )
```

příklad: (liche 123) -> t
příklad: (sude 123) -> nil

Rotace seznamu:

```
(defun rotace(s)
  (reverse (cons (car s) (reverse (cdr s))))
)

(defun rotace_zpet(s)
  (cons (car (reverse s)) (reverse (cdr (reverse s))))
)
```

příklad: (rotace '(a b c d)) -> (b c d a)
příklad: (rotace_zpet '(a b c d)) -> (d a b c)

Substituce:

a) 1.výskytu v jednoúrovňovém (lineárním) seznamu
b) všech výskytů v jednoúrovňovém (lineárním) seznamu
c) všech výskytů ve víceúrovňovém seznamu

```
(defun substituce1(a b s)
  (cond ((null s) nil)
        ((equal a (car s)) (cons b (cdr s)))
        (t (cons (car s) (substituce1 a b (cdr s))))))

(defun substituce2(a b s)
  (cond ((null s) nil)
        ((equal a (car s)) (cons b (substituce2 a b (cdr s))))
        (t (cons (car s) (substituce2 a b (cdr s))))))

(defun substituce3(a b s)
  (cond ((null s) nil)
        ((listp (car s)) (cons (substituce3 a b (car s)) (substituce3 a b (cdr s))))
        ((equal a (car s)) (cons b (substituce3 a b (cdr s))))
        (t (cons (car s) (substituce3 a b (cdr s))))))
```

příklad: (substituce1 'a 'b '((a b) a (c (d a)) a)) -> ((a b) b (c (d a)) a)
příklad: (substituce2 'a 'b '((a b) a (c (d a)) a)) -> ((a b) b (c (d a)) b)
příklad: (substituce3 'a 'b '((a b) a (c (d a)) a)) -> ((b b) b (c (d b)) b)

Lisp 3-4

Vyjmutí prvku:

a) 1.výskytu v jednoúrovňovém (lineárním) seznamu
b) Všech výskytů v jednoúrovňovém (lineárním) seznamu
c) Všech výskytů ve víceúrovňovém seznamu

```
(defun vyjmout1(a  s)
  (cond ((null s) nil)
        ((equal a (car s)) (cdr s))
        (t (cons (car s) (vyjmout1 a  (cdr s))))))

(defun vyjmout2(a  s)
  (cond ((null s) nil)
        ((equal a (car s)) (vyjmout2 a  (cdr s)))
        (t (cons (car s) (vyjmout2 a  (cdr s))))))

(defun vyjmout3(a  s)
  (cond ((null s) nil)
        ((listp (car s)) (cons (vyjmout3 a  (car s)) (vyjmout3 a  (cdr s))))
        ((equal a (car s)) (vyjmout3 a  (cdr s)))
        (t (cons (car s) (vyjmout3 a  (cdr s))))))
```

příklad: (vyjmout1 'a '((a b) a (c (d a)) a)) -> ((a b) (c (d a)) a)
příklad: (vyjmout2 'a '((a b) a (c (d a)) a)) -> ((a b) (c (d a)))
příklad: (vyjmout3 'a '((a b) a (c (d a)) a)) -> ((b) (c (d)))