Lisp 7-1

opakování destruktivních změn seznamů

Seznam bez posledního prvku - destruktivně:
;velmi důležité si uvědomit jak to funguje .. zejména návratová hodnota funkce
```
(defun bez_posledniho(s)
  (cond ((null s) nil)
        ((null (cdr s)) nil)
        ((null (cdr (cdr s))) (rplacd s nil) s)
        (t (bez_posledniho (cdr s)) s)
) )
```
příklad:
(setq s '(a b c d))
(bez_posledniho s)
s -> (a b c)

Přidá prvek na konec seznamu:
```
(defun pridat_na_konec(a s)
   (rplacd (last s) (cons a nil))
   s                                   ;návratová hodnota funkce
)
```
příklad:
(setq s '(a b c d))
(pridat_na_konec 'e s) -> (a b c d e)
s -> (a b c d e)

Rotace seznamu doleva:
```
(defun rotace_doleva(s)
  (cond ((null s) nil)
        (t (pridat_na_konec (car s) s)      ;změna seznamu
           (cdr s)                          ;návratová hodnota funkce
))      )
```
příklad:
(setq s '(a b c d))
(rotace_doleva s) -> (b c d a)
s -> (a b c d a)

```
(defun rotace_doleva2(s)
  (cond ((null s) nil)
        (t (pridat_na_konec (car s) s) ;1. změna seznamu
           (rplaca s (cadr s))         ;2. změna seznamu
           (rplacd s (cddr s))         ;3. změna seznamu
           s                           ;návratová hodnota
) )     )
```
příklad:
(setq s '(a b c d))
(rotace_doleva2 s) -> (b c d a)
s -> (b c d a)

Lisp 7-2

Rotace seznamu doprava:

```
(defun rotace_doprava(s)
  (cond ((null s) nil)
        (t (let ((posledni (last s))
                 (bez_posledniho (bez_posledniho s)))
                 (cons (car posledni) bez_posledniho)
) )    ) )
```

příklad:

```
(setq s '(a b c d))
(rotace_doprava s) -> (d a b c)
s -> (a b c)
```

```
(defun rotace_doprava2(s)
  (cond ((null s) nil)
        (t
          (let ((posledni (last s)))
               (bez_posledniho s)
               (let ((nova_bunka (cons (car s) (cdr s))))
                 (rplaca s (car posledni))
                 (rplacd s nova_bunka)
                 s
) ) )   )   )   )
```

příklad:

```
(setq s '(a b c d))
(rotace_doprava2 s) -> (d a b c)
s -> (d a b c)
```

Vložení prvku do seznamu na danou pozici:

```
(defun insert(prvek index seznam)
  (cond ((zerop index) (let ((nova_bunka (cons (car seznam) (cdr seznam))))
                             (rplaca seznam prvek)
                             (rplacd seznam nova_bunka)))

        ((eql index 1) (let ((nova_bunka (cons prvek (cdr seznam))))
                             (rplacd seznam nova_bunka)))

        ((null seznam) nil)
        (t (insert prvek (- index 1) (cdr seznam)))
) )
```

příklad:

```
(setq s '(a b c d))
(insert 'e 2 s)
s -> (a b e c d)

(setq s '(a b c d))
(insert 'e 4 s)
s -> (a b c d e)

(setq s '(a b c d))
(insert 'e 0 s)
s -> (e a b c d)
```

Lisp 7-3

Obrácení pořadí prvků v seznamu - destruktivně uvnitř:

```
(defun obrat(s)
  (obrat2 s nil))
```

Obrácení pořadí prvků v seznamu - destruktivně i navenek:

```
(defun obrat_seznam(s)
  (let ((obraceny (obrat2 (cons (car s) (cdr s)) nil)))
    (rplaca s (car obraceny))
    (rplacd s (cdr obraceny))
) )

(defun obrat2(zdroj druhy)
   (cond ((null zdroj) druhy)
         (t (let ((zbytek (cdr zdroj)))
              (rplacd zdroj druhy)
              (obrat2 zbytek zdroj)
) )     ) )
```

příklad:

```
(setq s '(a b c d))
(obrat s) -> (d c b a)
s -> a

(setq s '(a b c d))
(obrat_seznam s) -> (d c b a)
s -> (d c b a)
```

Substituce:

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

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

příklad:

```
(setq s '((a b) a (c (d a)) a))
(substituce1 'a 'b s) -> (b (c (d a)) a) ;na návratovou hodnotu jsme nijak nehráli
s -> ((a b) b (c (d a)) a)

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

příklad:

```
(setq s '((a b) a (c (d a)) a))
(substituce2 'a 'b s) -> nil   ;na návratovou hodnotu jsme nijak nehráli
s -> ((a b) b (c (d a)) b)
```

Lisp 7-4

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

příklad:
(setq s '((a b) a (c (d a)) a))
(substituce3 'a 'b s) -> nil
s -> ((b b) b (c (d b)) b)

```
(defun substituce4 (a b s)
  (declare (special first))
  (setq first t)
  (substituce4_1 'first a b s)
)

(defun substituce4_1 (first a b s)
  (cond ((null s) nil)
    ((and (equal a (car s)) (eval first)) (set first nil) (rplaca s b) (substituce4_1 first a b (cdr
s)))
    ((listp (car s)) (substituce4_1 first a b (car s)) (substituce4_1 first a b (cdr s)))
    (t (substituce4_1 first a b (cdr s)))
) )
```

příklad:
(setq s '((a b) a (c (d a)) a))
(substituce4 'a 'b s) -> nil
s -> ((b b) a (c (d a)) a)

Odstranit:

a) 1.výskyt v jednoúrovňovém (lineárním) seznamu
b) vechny výskyty v jednoúrovňovém (lineárním) seznamu
c) vechny výskyty ve víceúrovňovém seznamu
d) 1.výskyt ve víceúrovňovém seznamu

```
(defun odstranit1(a s)
  (cond ((null s) nil)
        ((and (equal a (car s)) (null (cdr s))) nil)
        ((equal a (car s))
            (let ((nova_bunka (cons (cadr s) (cddr s))))
                (rplaca s (car nova_bunka))
                (rplacd s (cdr nova_bunka))
                s))
        ((equal a (cadr s)) (rplacd s (cddr s)) s)
        (t (odstranit1 a (cdr s)) s)
) )
```

Lisp 7-5
příklad:
(setq s '(a (a b) a (c (d a)) a))
(odstranit1 'a s) -> ((a b) a (c (d a)) a)
s -> ((a b) a (c (d a)) a)
(setq s '(a))
(odstranit1 'a s) -> nil
s -> (a)

```
(defun odstranit2(a s)
  (cond ((null s) nil)
        ((equal a (car s)) (odstranit2 a (cdr s)))
        ((equal a (cadr s)) (rplacd s (cddr s)) (odstranit2 a (cdr s)) s)
        (t (odstranit2 a (cdr s)) s)
) )
```

příklad:
(setq s '(a (a b) a (c (d a)) a))
(odstranit2 'a s) -> ((a b) (c (d a)))

```
(defun odstranit2_1(a s)
  (cond ((null s) nil)
        ((equal a (car s)) (odstranit2_1 a (cdr s)))
        ((equal a (cadr s)) (rplacd s (odstranit2_1 a (cddr s))) s)
        (t (rplacd s (odstranit2_1 a (cdr s))) s)
) )
```

příklad:
(setq s '(a (a b) a (c (d a)) a))
(odstranit2_1 'a s) -> ((a b) (c (d a)))

```
(defun odstranit3(a s)
  (cond ((null s) nil)
        ((listp (car s)) (rplaca s (odstranit3 a (car s)))
                         (rplacd s (odstranit3 a (cdr s))))
        ((and (equal a (car s)) (null (cdr s))) nil)
        ((equal a (car s)) (rplaca s (cadr s))
                           (rplacd s (cddr s))
                           (odstranit3 a s)
                           s)
        (t (rplacd s (odstranit3 a (cdr s))) s)
) )

(defun odstranit3(a s)
  (cond ((null s) nil)
        ((equal a (car s)) (odstranit3 a (cdr s)))
        ((listp (car s)) (rplaca s (odstranit3 a (car s)))
                         (cond ((equal a (cadr s))
                                (rplacd s (odstranit3 a (cddr s))) s)
                                (t (rplacd s (odstranit3 a (cdr s))) s))
        )                )
        (t (rplacd s (odstranit3 a (cdr s))) s)
) )
```

příklad:
(setq s '(a (a b) a (c (d a)) a))
(odstranit3 'a s) -> ((b) (c (d)))

```
(defun odstranit4 (a s)
  (declare (special first))
  (setq first t)
  (odstranit4_1 'first a s)
)

(defun odstranit4_1 (first a s)
   (cond ((null s) nil)
        ((and (equal a (car s)) (eval first)) (set first nil) (cdr s))
        ((listp (car s)) (rplaca s (odstranit4_1 first a (car s)))
                    (cond ((and (equal a (cadr s)) (eval first))
                           (set first nil) (rplacd s (cddr s)) s)
                          (t (rplacd s (odstranit4_1 first a (cdr s))))
         )          )
        (t (rplacd s (odstranit4_1 first a (cdr s))) s)
) )
```

příklad:
(setq s '((a b) a (c (d a)) a))
(odstranit4 'a s) -> ((b) a (c (d a)) a))
(setq s '(a (a b) a (c (d a)) a))
(odstranit4 'a s) -> ((a b) a (c (d a)) a))
(setq s '((b c) d (c (d a)) a))
(odstranit4 'a s) -> ((b c) d (c (d)) a))
(setq s '(b c a (c (d a)) a))
(odstranit4 'a s) -> (b c (c (d a)) a)

Aplikace nad strukturovanými seznami

Seznam rodičů:
reprezentace: (jmeno matka otec)

```
(defun jmeno(zaznam)
  (car zaznam))

(defun matka(zaznam)
  (cadr zaznam))

(defun otec(zaznam)
  (caddr zaznam))

(defun rodice(zaznam)
  (cdr zaznam))

(defun vytvor(jmeno matka otec)
  (list jmeno matka otec))
```

Lisp 7-7

Vrátí rodiče k danému jménu:

```
(defun najdi_rodice(jm seznam)
  (declare (special jm))
  (mapcan '(lambda(zaznam)
                  (cond ((eq jm (jmeno zaznam)) (rodice zaznam))
                        (t nil)
            )     )  seznam))
```

příklad:

```
(setq seznam_rodicu
     '((jan dana zdenek)
       (dana jirina antonin)
       (milos jirina antonin)
       (petr milos jitka)
       (marek zdenek petra)
       (martina dana michal)
      ))
```

(najdi_rodice 'petr seznam_rodicu) -> (milos jitka)

Najde všechny babicky:

```
(defun babicky(jm seznam)
  (let ((rod (najdi_rodice jm seznam)))
    (cond ((null rod) nil)
          (t (mapcan '(lambda (x)
                        (cond ((null x) nil)
                              ((list (car x)))
                      )     )
                      (list (najdi_rodice (car rod) seznam)
                            (najdi_rodice (cadr rod) seznam))
) ) )   )  )
```

příklad: (babicky 'jan seznam_rodicu) -> (jirina)

Najde všechny dědečky:

```
(defun dedecci(jm seznam)
  (let ((rod (najdi_rodice jm seznam)))
     (cond ((null rod) nil)
           (t (append (cdr (najdi_rodice (car rod) seznam))
                      (cdr (najdi_rodice (cadr rod) seznam)))
) ) )    )
```

příklad: (dedecci 'jan seznam_rodicu) -> (antonin)

Najde všechny předky:

```
(defun predci(jm seznam)
  (let ((rod (najdi_rodice jm seznam)))
    (cond ((null rod) nil)
          (t (append rod (predci (car rod) seznam) (predci (cadr rod) seznam)))
)) )
```

;příklad: (predci 'jan seznam_rodicu) -> (DANA ZDENEK JIRINA ANTONIN)
;příklad: (predci 'petr seznam_rodicu) -> (MILOS JITKA JIRINA ANTONIN)

Lisp 7-8

Najde vechny vlastní sourozence:

```
(defun sourozenci(jm seznam)
  (declare (special jm))
  (let ((rod (najdi_rodice jm seznam)))
    (declare (special rod))
    (cond ((null rod) nil)
          (t (mapcan '(lambda(zaznam)
                        (cond ((and (equal rod (rodice zaznam))
                                    (not (equal jm (jmeno zaznam)))
                                    ) (list (jmeno zaznam)))
                              (t nil)
                        )     )
                     seznam)
) ) )   )
```

příklad: (sourozenci 'jan seznam_rodicu) -> nil
příklad: (sourozenci 'dana seznam_rodicu) -> (milos)
příklad: (sourozenci 'milos seznam_rodicu) -> (dana)

Najde i nevlastní sourozence:

```
(defun sourozenci2(jm seznam)
  (declare (special jm))
  (let ((rod (najdi_rodice jm seznam)))
    (declare (special rod))
    (cond ((null rod) nil)
          (t (mapcan '(lambda(zaznam)
                        (cond ((and (or (member (car rod) (rodice zaznam))
                                        (member (cadr rod) (rodice zaznam))
                                    )
                                    (not (equal jm (jmeno zaznam)))
                                    ) (list (jmeno zaznam)))
                              (t nil)
                        )     )
                     seznam)
) ) )   )
```

příklad: (sourozenci 'jan seznam_rodicu) -> nil
příklad: (sourozenci2 'jan seznam_rodicu) -> (marek martina)
příklad: (sourozenci2 'dana seznam_rodicu) -> (milos)
příklad: (sourozenci2 'milos seznam_rodicu) -> (dana)

Vrátí jméno k daným rodičům:

```
(defun najdi_jmeno(rod seznam)
  (declare (special rod))
  (mapcan '(lambda(zaznam)
             (cond ((equal rod (rodice zaznam)) (list (jmeno zaznam)))
                   (t nil)
             )    )
          seznam)
)
```

Zplošti seznam:

```
(defun zplosti(s)
  (cond ((null s) nil)
        ((atom s) s)
        ((atom (car s)) (cons (car s) (zplosti (cdr s))))
        (t (append (zplosti (car s)) (zplosti (cdr s))))
) )

(defun zplosti2(s)
  (mapcan '(lambda(a)
                  (cond ((atom a) (list a))
                        (t (zplosti2 a))
             )      )
          s)
)
```

příklad: (zplosti '(a ((b c) d) (e (f)))) -> (a b c d e f)
příklad: (zplosti2 '(a ((b c) d) (e (f)))) -> (a b c d e f)

Vlastnosti symbolických atomů

- jméno atomu je pouze jednou z jeho možných vlastností, patří mezi standardní vlastnosti
- každému sybolickému atomu můžeme přidat libovolný počet dalších vlastností
- reprezentace vlastností - objet pomocí zřetězeného seznamu cons-buněk
- seznam vlastnostní seznamu nelze chápat jako obyčejný seznam, neboť je součástí vnitřní reprezentace atomu
- vlastnosti tvoří základ pro programy řízené daty a objektové programování v Lispu

(put atom vlastnost hodnota)	.. nastavení hodnoty
(get atom vlastnost)	.. získání hodnoty
(remprop atom vlastnost)	.. zrušení vlastnosti
(symbol-plist symbol)	.. seznam vlastností a jejich hodnot

příklad:
(symbol-plist 'a) -> nil
(get 'a 'vl1) -> nil
(put 'a 'vl1 1)
(symbol-plist 'a) -> (vl1 1)
(put 'a 'vl2 2)
(symbol-plist 'a) -> (vl2 2 vl1 1)
(get 'a 'vl1) -> 1
(remprop 'a 'vl1) -> t
(get 'a 'vl1) -> nil
(symbol-plist 'a) -> (vl2 2)

Lisp 7-10

Seznam rodičů pomocí vlastností:

```
(put 'jan 'rodice '(dana zdenek))
(put 'dana 'rodice '(jirina antonin))
(put 'milos 'rodice '(jirina antonin))
(put 'petr 'rodice '(milos jitka))
(put 'marek 'rodice '(zdenek petra))
(put 'martina 'rodice '(dana michal))


(defun najdi_rodice(jm)
  (get jm 'rodice))

(defun predci(jm)
   (let ((rod (najdi_rodice jm)))
     (cond ((null rod) nil)
           (t (append rod (predci (car rod)) (predci (cadr rod))))
) ) )
```

příklad: (predci 'jan) -> (DANA ZDENEK JIRINA ANTONIN)

Asociativní vyhledávání

- assoc v seznamu vyhledá podseznam pod car prvku

příklad:

(assoc 'elf '((a 0 1) (b d x 3 x) (elf 23 alfa)) -> (elf 23 alfa)