Úloha 1

- definujte obecnou funkci (**seznamator** kombi L1 L2), která dostane kombinační funkci a dva seznamy, vrátí seznam tvořený výsledky kombinační funkce pro dvojice prvků na stejných pozicích v L1 a L2
- definujte její pomocí funkce
  - (soucet L1 L2), která sčítá dvojice prvků
  - (mensi L1 L2), která vybírá menší z dvojice prvků

Řešení 1

```
(define (seznamator kombi L1 L2)
    (cond
        [(or (empty? L1) (empty? L2)) '()]
        [else
            (cons (kombi (car L1) (car L2))
                  (seznamator kombi (cdr L1) (cdr L2)))]
  ))
(define (soucet L1 L2) (seznamator + L1 L2))
(define (mensi L1 L2) (seznamator min L1 L2))
```

Úloha 2

- definujte obecnou vyhledávací funkci pro seznamy (**hledej** shoda? hodnota vysledek L), která dostane kritérium (shoda?) pro porovnání prvku s cílovou hodnotou a způsob určení výsledku z nalezeného prvku seznamu (vysledek)
- definujte její pomocí funkce pro vyhledání telefonního čísla podle jména a jména podle telefonního čísla v seznamu kontaktů (define-struct kontakt (jmeno telefon))

Řešení 2 (1)

```
(define (hledej shoda? hodnota vysledek L)
    (cond
        [(empty? L) #f]
        [(shoda? hodnota (car L)) (vysledek (car L))]
        [else (hledej shoda? hodnota vysledek (cdr L))]
))
```

Řešení 2 (2)

```
(define (hledej-cislo jmeno telseznam)
    (local (
        (define (jmeno=? x y)
            (string=?    x
                         (kontakt-jmeno y)))
    )
    (hledej jmeno=? jmeno kontakt-telefon telseznam)
    ))
```

Řešení 2 (3)

```
(define (hledej-jmeno cislo telseznam)
    (local (
        (define (cislo=? x y)
            (=  x
                (kontakt-telefon y)))
    )
    (hledej cislo=? cislo kontakt-jmeno telseznam)
    ))
```

Úloha 3

- definujte generickou funkci (**vyberBVS** podminka? strom), která projde binární vyhledávací strom a vrátí seznam všech klíčů z něj, které vyhoví zadané podmínce
- její pomocí vyberte z binárního vyhledávacího stromu všechna sudá čísla

Řešení 3

```
(define (vyberBVS podminka? strom)
    (if (empty? strom) '()
        (append
            (vyberBVS podminka? (uzelBVS-levy strom))
            (if (podminka? (uzelBVS-klic strom))
                (list (uzelBVS-klic strom))
                '())
            (vyberBVS podminka? (uzelBVS-pravy strom))
        )))

(vyberBVS even? strom)
```

Úloha 4

- vytvořte vyhledávací funkci **hledejBVS?** pro generický binární vyhledávací strom, kde klíče mohou být libovolného typu
- funkce vrací true/false podle toho, zda hledaná hodnota je ve stromu obsažena nebo ne

Řešení 4

```
(define (hledejBVS? hodnota mensi? vetsi? strom)
    (cond
        [(empty? strom) #f]
        [(mensi? hodnota (uzelBVS-klic strom))
            (hledejBVS? hodnota mensi? vetsi?
                                    (uzelBVS-levy strom))]
        [(vetsi? hodnota (uzelBVS-klic strom))
            (hledejBVS? hodnota mensi? vetsi?
                                    (uzelBVS-pravy strom))]
        [else #t]
    ))
```

Úloha 5

- implementujte řazení slučováním (mergesort kriterium? L), které seřadí hodnoty v L vzestupně, pro porovnání dvojice hodnot se používá kriterium?
- řazení slučováním rozdělí seznam na dvě poloviny, ty seřadí a následně seřazené poloviny spojí do jednoho seřazeného seznamu

Řešení 5 (1)

```
(define (merge L1 L2)      ;spojí dva seřazené seznamy
   (cond
      [(empty? L1) L2]
      [(empty? L2) L1]
      [(<= (car L1) (car L2))
                  (cons    (car L1)
                           (merge (cdr L1) L2))]
      [else       (cons    (car L2)
                           (merge L1 (cdr L2)))]
   ))
```

Řešení 5 (2)

```
(define (split cast1 cast2 L) ;rozdělí seznam na poloviny
    (cond
        [(empty? L) (list cast1 cast2)]
        [(empty? (cdr L)) (list (append L cast1) cast2)]
        [else
            (split    (cons (car L) cast1)
                      (cons (second L) cast2)
                      (cdr (cdr L)))]
    ))
```

Řešení 5 (3)

```
(define (mergesort L)
    (cond
        [(empty? L) '()]
        [(empty? (cdr L)) L]
        [else (local (
                    (define poloviny (split '() '() L))
                )
                (merge  (mergesort (car poloviny))
                        (mergesort (second poloviny))))]
    ))
```

Řešení 5 (3)

```
(define (mergesort L)
    (local (
        (define (merge L1 L2) ... )
        (define (split cast1 cast2 L) ... )
        (define (mergesort L) ... )
    )
    (mergesort L)))
```