Úloha 1

- vytvořte funkci (**zip** L1 L2), která spojí seznamy L1 a L2 „zipem“ – ve výsledku se budou střídat hodnoty z obou seznamů
- (zip '(10 30 5 17) '(2 14 6)) vrací (10 2 30 14 5 6 17)

Řešení 1

```
(define (zip L1 L2)
    (if (empty? L1) L2
            (cons (car L1)
                    (zip L2 (cdr L1)))
    )
)
```

Úloha 2

- vytvořte funkci (**sectivetsi** L1 L2), která dostane dva seznamy čísel, jejím výsledkem je seznam, který pro každé číslo X z L1 obsahuje součet všech čísel z L2 větších než X
- např. (sectivetsi '(15 8 35) '(10 20 30 5)) vrací (50 60 0)

  čísla z L2 větší než 15 jsou 20 a 30, součet je 50

- tip: vytvořte si pomocnou funkci počítající součet čísel větších než X

Řešení 2

```
(define (sectivetsi L1 L2)
   (if (empty? L1) '()
         (cons (soucetvetsich (car L1) L2)
               (sectivetsi (cdr L1) L2))))

(define (soucetvetsich X L)
   (cond
      [(empty? L) 0]
      [(> (car L) X) (+ (car L) (soucetvetsich X (cdr L)))]
      [else (soucetvetsich X (cdr L))]))
```

Úloha 3

- stáhněte si https://www.nti.tul.cz/~satrapa/vyuka/altprog/bvs-data.rkt – obsahuje datovou strukturu pro uzel binárního vyhledávacího stromu (BVS)
  (define-struct **uzelBVS** (klic levy pravy))
  a konstantu testovaci obsahující takový strom
- definujte funkci (**hloubka** strom), která spočítá hloubku daného BVS (délku nejdelší cesty z kořene do listu)

Řešení 3

```
(define (hloubka strom)
    (if (empty? strom) 0
            (+ 1   (max (hloubka (uzelBVS-levy strom))
                        (hloubka (uzelBVS-pravy strom))))
    )
)
```

Úloha 4

- vytvořte funkci
  (**linearizuj** strom)
  která vrátí seznam všech klíčů obsažených v BVS
- procházejte strom v infixovém pořadí (nejprve levý podstrom, pak klíč a po něm pravý podstrom), výsledný seznam pak bude seřazen

Řešení 4

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

Úloha 5

- vytvořte funkci (**vyvazeni** strom), která vrátí největší rozdíl mezi hloubkou pravého a levého podstromu uzlů daného stromu

Řešení 5

```
(define (vyvazeni strom)
    (if (empty? strom) 0
        (max
            (abs (-   (hloubka (uzelBVS-levy strom))
                      (hloubka (uzelBVS-pravy strom))))
            (vyvazeni (uzelBVS-levy strom))
            (vyvazeni (uzelBVS-pravy strom))
        )
    )
)
```

Úloha 6

- vytvořte funkci (**jeBVS?** strom), která zkontroluje, zda je strom binární vyhledávací, tedy zda pro každý jeho uzel platí, že klíče v jeho levém podstromu jsou menší a v pravém větší než klíč uzlu

Řešení 6

```
(define (jeBVS? strom)
   (if  (empty? strom)  #t
      (and
         (mensi? (uzelBVS-klic strom)
                        (uzelBVS-levy strom))
         (vetsi? (uzelBVS-klic strom)
                        (uzelBVS-pravy strom))
         (jeBVS? (uzelBVS-levy strom))
         (jeBVS? (uzelBVS-pravy strom)))
   ))
```

Řešení 6

```
(define (mensi? N strom)
    (cond
        [(empty? strom) #t]
        [(<= N (uzelBVS-klic strom)) #f]
        [else
            (and  (mensi? N (uzelBVS-levy strom))
                  (mensi? N (uzelBVS-pravy strom)))]
    )
)
```

Řešení 6 – kreativní verze

```
(define (jeBVS? strom)
   (usporadany? (linearizuj strom)))

(define (usporadany? L)
   (cond
      [(empty? L) #t]
      [(empty? (cdr L)) #t]
      [else (and  (< (car L) (second L))
                  (usporadany? (cdr L)))]))
```