Paradigmata programovaní 1

Instrukce:

- Řešení v podobě zdrojového kódu ve Scheme zasílejte v jediném souboru na e-mail

  `eduard.bartl@upol.cz`.

- Jako předmět e-mailové zprávy uveďte

  `PAPR1 -- reseni ukolu c. 3`.

- Do zdrojového kódu nepište ani příklady užití uvedené v zadání ani Vaše testy.
- Soubor se zdrojovým kódem pojmenujte jako

  `UzivatelskeJmeno.ss` (nebo `UzivatelskeJmeno.rkt`),

  kde `UzivatelskeJmeno` je uživatelské jméno ve Vaší univerzitní e-mailové adrese.

Upozornění: Nedodržení instrukcí může znamenat neuznání celého úkolu.

Úkol č. 3

datum zadaní: 4. prosince 2012
termín odevzdání: 12. prosince 2012

1. **(4 body)** Pomocí `foldl` nebo `foldr` naprogramujte proceduru `count` zjišťující počet výskytů prvků v daném seznamu.

   Příklady použití:

   ```
   > (count '(3 1 3 2 1 2 3 3 3))
   ((3 . 5) (1 . 2) (2 . 2))

   > (count '(d b a c b b a))
   ((d . 1) (b . 3) (a . 2) (c . 1))
   ```

2. **(4 body)** Pomocí rekurze naprogramujte proceduru `map-index-pred`. Tuto proceduru znáte z předchozího úkolu. Přijímá tři parametry: predikát $\triangleleft pred?\triangleright$, procedura $\triangleleft f\triangleright$ a seznam $\triangleleft l\triangleright$ = $(a_1\ a_2\ \ldots a_n)$. Procedura `map-index-pred` vrací seznam $(b_1\ b_2\ \ldots b_n)$ stejné délky jako $\triangleleft l\triangleright$, jeho prvky b_i jsou $f(a_i)$ pro indexy splňující predikát $\triangleleft pred?\triangleright$ a a_i pro indexy nesplňující predikát $\triangleleft pred?\triangleright$. Např:

   ```
   > (map-index-pred odd? sqr '(2 3 4 5))
   (2 9 4 25)
   ```

   Prvky seznamu na lichých indexech – tedy prvky 3 a 5 – jsou ve výsledném seznamu umocněnny. Protože indexy prvků 2 a 4 jsou 0 a 2, tedy nejsou liché, jsou tyto prvky ve výsledném seznamu stejné jako v původním. Další příklad:

   ```
   > (map-index-pred (lambda(i) (< i 2)) - '(1 2 3 4 5))
   (-1 -2 3 4 5)
   ```

   Prvky na indexech menší než 2 jsou ve výsledném seznamu nahrazeny jejich opačnou hodnotou, ostatní zůstavají stejné.

3. **(7 bodů)** Napište proceduru přijímající jeden argument $\triangleleft n\triangleright$, která vrací $(n+1)$-ní člen C_{n+1} Catalanovy posloupnosti. Catalanova posloupnost (používaná mj. v kombinatorice) je daná předpisem:

$$C_1 = 1,$$
$$C_{n+1} = \frac{2(2n+1)}{n+2} C_n \text{ pro } n > 0.$$

Proceduru implementujte
(a) jako rekurzivní proceduru (která není koncově rekurzivní); tuto proceduru pojmenujte `catalan1`,
(b) jako iterativní proceduru, kde je pro modifikaci lokálního prostřdí využita speciální forma `define`; tuto proceduru pojmenujte `catalan2`,
(c) jako iterativní proceduru pomocí pojmenovaného `let`; tuto proceduru pojmenujte `catalan3`.

Příklad použití:

```
> (catalan1 1)
2
```

4. **(7 bodů)** Naprogramujte proceduru, která vypočte největší společný dělitel dvou zadaných čísel pomocí Euklidova algoritmu. Proceduru implementujte
(a) jako rekurzivní proceduru (která není koncově rekurzivní); tuto proceduru pojmenujte `euclid1`,
(b) jako iterativní proceduru, kde je pro modifikaci lokálního prostřdí využita speciální forma `define`; tuto proceduru pojmenujte `euclid2`,
(c) jako iterativní proceduru pomocí pojmenovaného `let`; tuto proceduru pojmenujte `euclid3`.

Příklady použití:

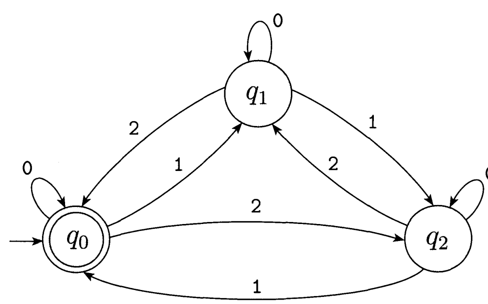
```
> (euclid1 9 24)
3

> (euclid2 17 25)
1

> (euclid3 5 5)
5
```

5. **(3 bodů)** Napište iterativní proceduru `harmonic-mean` vracející harmonický průměr čísel v seznamu. Harmonický průměr H reálných čísel $x_1, x_2, \ldots, x_n$ je dán vztahem:

$$H = \frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}}.$$

Při implementaci je zakázáno použít proceduru `length`.

6. **(7 bodů)** Napište iterativní proceduru `divided-by-three?`, která zjistí, jestli je součet čísel v daném seznamu dělitelný třemi. K implementaci použijte tzv. *konečný automat*, jehož činnost je popsána orientovaným grafem znázorněným na následujícím obrázku:



Vrcholy označené jako q_0, q_1, q_2 představují stavy, ve kterých se automat může nacházet, hrany znázorňují do jakého stavu automat přejde zpracováním čísla určeného ohodnocením dané hrany. Činnost automatu je podrobněji popsána na následujícím příkladu:

```
> (divided-by-three? '(2 0 2 1 1))
#t
```

Stav automatu na začátku zpracování seznamu je vždy q_0 (vrchol označený šipkou, která vede ,,odnikud"). Automat může zpracovávat seznam čísel např. zleva doprava, tzn. začne číslem 2. Přejde tedy ze stavu q_0 do stavu q_2 (díky hraně, která směřuje z q_0 do q_2 a je ohodnocena číslem 2). Poté zpracováná další číslo v seznamu, tzn. číslo 0. Hrana ohodnocená nulou vycházející z vrcholu q_2 vede opět do vrcholu q_2, automat tedy zpracováním čísla 0 svůj stav nezmění. Dál je přečteno číslo 2, automat přejde do stavu q_1. Zpracováním čísla 1 přejde z q_1 zpět do q_2 a na závěr přečtením čísla 1 přejde do stavu q_0. Pokud bude stav automatu po přečtení všech čísel q_0[1], pak je součet všech čísel dělitelný třemi. Pokud bude jeho stav na konci zpracování jiný než q_0, pak součet není dělitelný třemi. Další príklad:

```
(divided-by-three? '(0 1 2 2 2 2 1))
#f
```

Automat po přečtení všech čísel skončí ve stavu q_1, tzn., že součet čísel $0 + 1 + 2 + 2 + 2 + 2 + 1$ není dělitelný třemi.

[1]Jedná se o tzv. koncový stav; typicky bývá označen dvojitým kroužkem. V tomto případě je náhodou počáteční a koncový stav automatu jeden a týž.