KMI/YPP1 Paradigmata programování 1
úkol č. 3

1. Vytvořte proceduru `singletons` očekávající jako argument seznam a vracející seznam singletonů (tzn. jednoprvkových seznamů) tvořených prvky vstupního seznamu.

   Příklady použití:

   ```
   > (singletons '(2))
   ((2))

   > (singletons '(2 -1 3))
   ((2) (-1) (3))
   ```

2. Vytvořte proceduru `roots-of-unity` s jedním argumentem ◁n▷, která vrátí seznam délky n obsahující všechny kořeny rovnice

   $$x^n = 1.$$

   Pro výpočet k-tého kořenu, kde $k = 0, 1, \ldots, n-1$, využijte známý vztah

   $$x_k = \cos\frac{2\pi k}{n} + i \cdot \sin\frac{2\pi k}{n},$$

   kde i je imaginární jednotka. Jak víme, kořenem může být komplexní číslo. Každý kořen upravte pomocí následující aplikace `(rationalize (inexact->exact x) 1/10)`, kde `x` je reálná nebo imaginární část kořene (viz str. 30 prvního dílu výukového textu).

   Příklady použití:

   ```
   > (roots-of-unity 2)
   (1 -1)

   > (roots-of-unity 6)
   (1 1/2+4/5i -1/2+4/5i -1 -1/2-4/5i 1/2-4/5i)
   ```

3. Vytvořte proceduru `div-list` s jedním argumentem ◁n▷, která vrátí seznam délky n obsahující pravdivostní hodnoty `#t` a `#f` podle toho, jestli číslo dané pozicí v seznamu dělí číslo n.

   Příklady použití:

   ```
   > (div-list 2)
   (#t #t)

   > (div-list 3)
   (#t #f #t)

   > (div-list 4)
   ```

```
(#t #t #f #t)

> (div-list 5)
(#t #f #f #f #t)

> (div-list 12)
(#t #t #t #t #f #t #f #f #f #f #f #t)
```

4. Naprogramujte proceduru `make-palindrom`, která vytváří z prvků vstupního seznamu palindrom. Je zakázáno použít procedury `reverse` a `append`.

   Příklady použití:

```
(make-palindrom '(a n))
> (a n n a)

(make-palindrom '(r o t))
> (r o t o r)

(make-palindrom '(d e n n i s))
> (d e n n i s s i n n e d)

(make-palindrom '(n e p o t))
> (n e p o t o p e n)
```

5. Naprogramujte proceduru `map-index-pred` se třemi argumenty: predikátem $\triangleleft pred? \triangleright$, procedurou $\triangleleft f \triangleright$ a seznamem $\triangleleft l \triangleright = (a_1\ \ a_2\ \ \ldots a_n)$. Procedura `map-index-pred` vrací seznam $(b_1\ \ b_2\ \ \ldots b_n)$, jehož 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 umocněny. Protože indexy prvků 2 a 4 jsou 0 a 2 (nejsou tedy liché), jsou tyto prvky ve výsledném seznamu stejné jako v původním seznamu. 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.