3.3. Exercícios de Fixação

3.3.1. Iteração

Resolva os seguintes exercícios utilizando "forms" para a implementação de laços iterativos: `do, loop, dolist`

1. Implemente uma função COMPRIMENTO que calcule de forma iterativa e devolva como resultado o número de elementos do primeiro nível de uma lista.

2. Escreva uma função INVERSO que, dada uma lista como parâmetro, devolva o inverso da lista. Assim para a Lista (A (B C) D) a função devolveria (D (B C) A).
   Toque: você pode usar a função append.

3. Escreva uma Função MEMBRO que, dado qualquer item (átomo ou lista) como seu primeiro parâmetro, devolva NIL se o primeiro parâmetro não aparecer no segundo, que deverá obrigatoriamente ser uma lista. Caso ele apareca, a função devolverá a lista restante, o ítem inclusive.
   Assim:
   (membro ´a ´(b c d)) = nil
   (membro ´c ´(a b c d e) = (c d e)

3.3.2. Recursão

Como já foi ressaltado em aula, a facilidade e a naturalidade com que a recursividade pode se utilizada em linguagens funcionais como LISP é um dos pontos fortes deste enfoque.

4. Reescreva a função MEMBRO acima de forma que seja uma função recursiva.

5. Escreva uma função recursiva INTERSECÇÂO que, dadas duas listas, devolva uma lista com os elementos comuns às duas.

3.3.3. Resposta a alguns exercícios

1.Comprimento/Iterativo

```
(defun comprimento (lista)
  (setq comp 0)
  (loop
     (when (eq lista NIL) (return comp))
     (setq comp (+ 1 comp))
     (pop lista)
  )
  comp
)
```

2. Inverso/Iterativo

```
(defun inverso (lista)
  (setq inversa '())
  (loop
      (setq inversa (push (pop lista) inversa))
      (when (eq lista nil) (return inversa))
  )
  inversa
)
```

4.Membro/Recursiva

```
(defun membroR (elemento lista)
   (if (equal nil lista)
       nil
       (if (equal elemento (car lista))
           lista
           ( membroR elemento (cdr lista))
       ) ;; fim se
   ) ;; fim se
) ;; fim membroR
```