Paradigmas de Linguagens de Programação
Lista 2
Prof. Sergio D. Zorzo

---

1. Um palíndromo é uma lista que tem a mesma sequência de elementos quando é lida tanto da esquerda para a direita quanto da direita para a esquerda. Defina uma função em LISP de nome PALINDROMIZE, que recebe uma lista como seu argumento e retorna um palíndromo com o dobro do comprimento. Defina também uma função auxiliar REVERSA, que recebe uma lista e retorna outra com os elementos em ordem inversa. (O interpretador LISP tem uma função pré-definida chamada REVERSE, que não deve ser usada neste exercício).

```
(defun reversa (lista)
         (cond ((null lista) '())
               ((null (cdr lista)) lista)
               (t (append (reversa (cdr lista)) (list (car lista)
      ))))
)

(defun palindromize (lista)
  (append lista (reversa lista)))
```

2. Defina um predicado PALINDROMOP, que testa seu argumento para ver se é um palindromo. Se o argumento for um átomo, a resposta deve ser NIL.
3. Defina um predicado TRI-RETAN, que recebe três argumentos. Os três argumentos são os comprimentos dos lados de um triângulo, que pode ser um triângulo retângulo. TRI-RETAN deve retornar T se a soma dos quadrados dos dois lados menores está a menos de 2% do quadrado do lado maior. Caso contrário, TRI-RETAN deve retornar NIL. Assuma que o lado maior é dado como o primeiro argumento.

```
(defun tri-retan (x y z)
  (let* ((somaQuadrados (+ (* y y) (* z z)))
         (quadradoMaior (* x x))
         (doisPorcento (* quadradoMaior 0.02)))
     (if (< (abs (- somaQuadrados quadradoMaior)) doisPorcento) t
      nil)
  )
)
```

4. Defina CIRCULO tal que retorne uma lista com a circunferência e a área de um círculo, dado o raio deste círculo. Assuma que PI é uma variável livre com o valor apropriado.
5. Descreva o que faz o seguinte procedimento:

```
(defun misterio (s)
       (cond ((null s) 1)
             ((atom s) 0)
             (t (max (add1 (misterio (car s)))
                           (misterio (cdr s)))
             )
       )
)
```

6. Descreva o que faz seguinte procedimento:

```
(defun estranho (l)
       (cond ((null l) nil)
             ((atom l ) l)
```

```
            (t (cons (estranho (car l))
                     (estranho (cdr l)))
            )
        )
)
```

7. Descreva o procedimento SQUASH, que recebe uma s-expressão como argumento e retorna uma lista simples com todos os átomos encontrados na s-expressão. Por exemplo:

```
> (SQUASH '(A (A (A (A B))) (((A B) B) B) B))
(A A A A B A B B B B)
```

8. Defina em LISP o predicado ESTA_EM, para dados um átomo e uma s-expressão, verificar se esse átomo está na s-expressão, em qualquer nível.

9. Descreva o que faz o seguinte procedimento:

```
(defun depth (exp)
       (cond ((null exp) 1)
             ((atom exp) 0)
             (t (+ apply 'max (mapcar 'depth exp)) 1))
       )
)
```

10. Defina um procedimento que toma como argumento uma lista de números e retorna a diferença entre o maior e o menor.

```
(defun maior (lista)
  (cond ((null lista) nil)
        ((null (cdr lista)) (car lista))
        ((> (car lista) (maior (cdr lista))) (car lista))
        (t (maior (cdr lista))))
)

(defun menor (lista)
  (cond ((null lista) nil)
        ((null (cdr lista)) (car lista))
        ((< (car lista) (menor (cdr lista))) (car lista))
        (t (menor (cdr lista))))
)

(defun diferenca (lista)
  (- (maior lista) (menor lista)))
```

11. Defina os procedimentos UNIAO, INTER, e DIFER para, dados dois conjuntos em forma de lista fazer as operações de união, intersecção, e diferença desses conjuntos, respectivamente.
12. Defina um predicado que testa se dois conjuntos tem ou não elementos em comum.
13. Defina um procedimento que testa se dois conjuntos representados como listas tem os mesmos elementos. Note que os elementos podem estar repetidos ou em ordem diferente.
14. Escreva um procedimento que, dados um elemento e uma lista, apaga todas as ocorrências desse elemento na lista.

```
(defun apaga (elemento lista)
       (cond ((null lista) nil)
             ((equal elemento (car lista))
                  (apaga elemento (cdr lista)))
             (t (cons (car lista) (apaga elemento (cdr lista))))))
)
```

15. Escreva um procedimento que, dados dois elementos e uma lista, sustitui todas as ocorrências do primeiro elemento pelo segundo, no primeiro nível da lista. Fazer uma versão recursiva e outra usando formas iterativas (do, dotimes, dolist, mapcar, etc).
16. Dadas duas listas representando conjuntos de elementos, construir um procedimento que faz o produto cartesiano (conjunto de pares ordenados) desses conjuntos. Fazer uma versão recursiva e outra usando formas iterativas (do, dotimes, dolist, mapcar, etc).
17. Defina uma função ECO que lê uma S-expressão e imprime a mesma S-expressão até ser lido o átomo FIM.