Exercícios - Laboratório 5

Enviar para ramon.medrado@inf.ufrgs.br com o assunto INF05008 - Turma X - Laboratório 5, onde $\mathbf{X} \in \{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$.

1. Desenvolva a função `cria-matriz` que, dados um *número natural positivo* n e uma lista contendo n^2 números, produz uma lista contendo n listas de n números cada. Exemplo:

```
(cria-matriz 2 (list 1 2 3 4))  deve produzir   (list (list 1 2)
                                                      (list 3 4))
```

2. Considere a seguinte definição de *web-page*.

```
;; Uma web-page é
;; - empty, ou
;; - (cons s wp), onde s:symbol e wp:web-page, ou
;; - (cons ewp wp), onde ewp:web-page e wp:web-page
```

   Um exemplo de web-page é

```
(define wp1
 (list 'The 'TeachScheme 'Web 'Page
       'Here 'you 'can 'find:
       (list 'LectureNotes 'for 'Teachers)
       (list 'Guidance 'for
             (list 'DrScheme: 'a 'Scheme 'programming 'environment))
       (list 'Exercise 'Sets)
       (list 'Solutions 'for 'Exercises)
       (list 'For 'further 'information
             (list 'email: 'sc@sc.com 'web: 'sc.com))))
```

   Um web-page que que ocorre dentro de outra é denominada *embedded web-page*. O nível de uma *embedded web-page* é o número de web-pages que a engloba.

   Desenvolva a função `palavras-niveis-impares` que, dada uma web-page, retorna a lista das palavras que ocorrem diretamente nas suas *embedded web-pages* de nível ímpar.

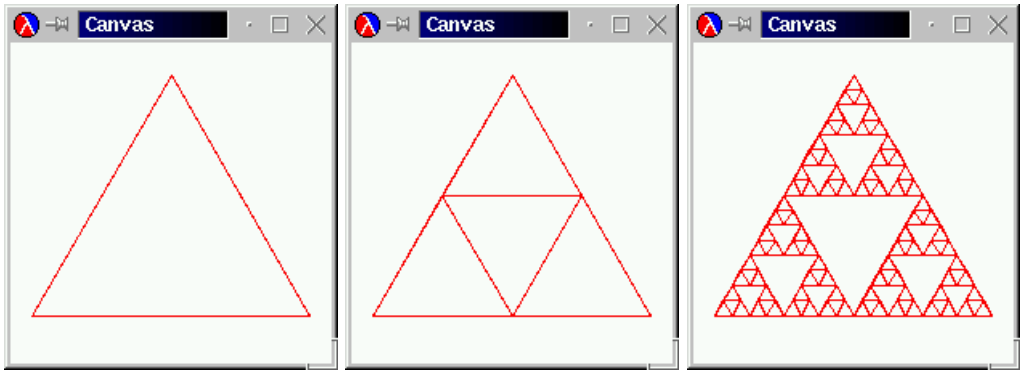
```
 (palavras-niveis-impares wp1) ==>
       (list 'LectureNotes 'for 'Teachers 'Guidance 'for 'Exercise 'Sets
             'Solutions 'for 'Exercises 'For 'further 'information)
```

3. Considere a seguinte definição de dados:

```
(define-struct nodo (filhos numeros))

;; Uma an (arvore-de-numeros) é uma estrutura
  (make-nodo f n), onde f:lista-de-an e n:lista-de-num

;; Uma lista-de-an é
;; - empty, ou
;; - (cons a l),  onde  a:an  e  l:lista-de-an

;; Uma lista-de-num
;; - empty, ou
;; - (cons n l), onde  n:number e l:lista-de-num
```

   Desenvolva a função `media-an`, que, dada uma árvore de números, retorna a média de todos os números contidos nesta.

4. As figuras abaixo mostram 3 etapas da construção de um fractal conhecido como *triângulo de Sierpinski*.



Para gerar esta figura, a idéia é ir dividindo um triângulo em 3 triângulos, e então cada um destes 3 é novamente dividido, e assim sucessivamente. A figura do meio mostra como seria o primeiro passo da divisão. A função que gera um triângulo de Sierpinski deve, a partir de um triângulo inicial, ir subdividindo-o recursivamente até que algum critério de fim seja atingido. Um possível critério seria que o tamanho dos triâgulos obtidos fossem pequenos demais para serem subdivididos. Complete a definição abaixo para desenhar triângulos de Sierpinski:

```
;; sierpinski : posn posn posn  ->  true
;; desenha um triângulo de Sierpinski com vértices nos pontos passados como
;; argumentos, e devolve true. Se triângulo passado como argumento tiver
;; dimensões muito pequenas, nada é desenhado.

(define (sierpinski a b c)
  (cond
    [(too-small? a b c) true]
    [else
      (local ((define a-b (mid-point a b))
              (define b-c (mid-point b c))
              (define c-a (mid-point a c)))
         (and
              (draw-triangle a b c)
              (sierpinski a a-b c-a)
              (sierpinski b a-b b-c)
              (sierpinski c c-a b-c)))]))

;; mid-point : posn posn  ->  posn
;; computa o ponto médio entre os 2 pontos passados como argumentos
;; exemplos: ...

(define (mid-point a-posn b-posn)
  (make-posn
    (mid (posn-x a-posn) (posn-x b-posn))
    (mid (posn-y a-posn) (posn-y b-posn))))



;; mid : number number  ->  number
;; computa a média entre os números passados como argumento
;; exemplos: ...

(define (mid x y)
  (/ (+ x y) 2))
```

```
;; draw-triangle : posn posn posn  ->  true
;;  ...

;; too-small? : posn posn posn  ->  bool
;; ...
```

Teste seu programa com o triângulo com vértices nos pontos A, B e C:

```
(define A (make-posn 200 0))
(define B (make-posn 27 300))
(define C (make-posn 373 300)
```

5. Para cada função *recursiva* desenvolvida nos exerícios anteriores (1 a 4), aponte se ela utiliza recursão estrutural ou recursão generativa. Se for recursão generativa, justifique apontando a chamada recursiva que não segue a estrutura de dados indutiva.

6. Para cada função *recursiva generativa* apontada no exercício anterior (5), responda às seguintes perguntas:

   a) Qual a instância trivial do problema?

   b) Qual a solução para a instância trivial?

   c) Como os subproblemas mais simples que o original são gerados? Quantos são gerados?

   d) Como as soluções dos subproblemas são combinadas para gerar a solução do problema original?

   e) Este programa sempre termina? Por quê?