INSTRUÇÕES: A lista de exercícios é para ser feita individualmente ou em duplas.

Desenvolva o quê é pedido abaixo, usando a estratégia de dividir para conquistar sempre que necessário.

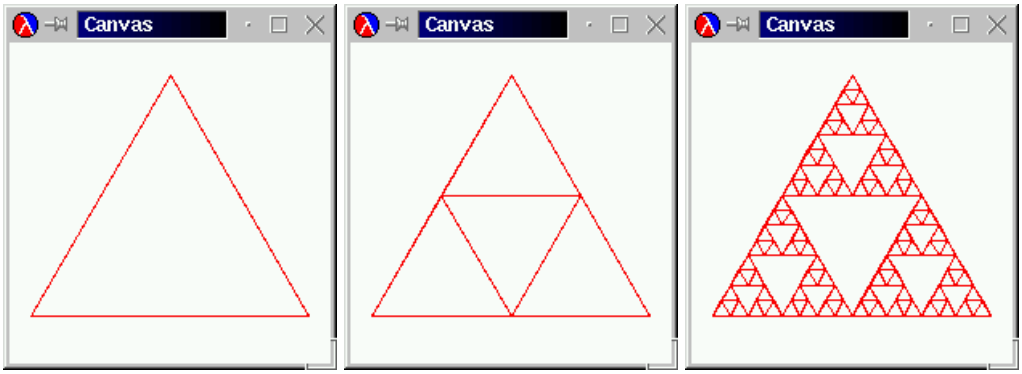
1. Considere a seguinte definição de dados:

   ```
   (define-struct nó (valor esq dir))
   ```

   Desenvolva a função `média-ab`, que, dada uma árvore binária, cujos nós são do tipo da estrutura `nó` definida acima, retorne a média de todos os valores contidos da árvore.

2. Dado o programa visto em aula que movimenta uma bola sobre uma mesa até que ela caia, modifique-o para que, agora, a mesa tenha uma borda que impeça a bola de cair. A borda não deve fazer a volta completa na mesa, cobrindo apenas 3 dos 4 lados desta. Dessa forma, se a bola chegar a um lado com borda, ela deve bater e voltar para dentro da mesa; caso contrário, ela deve cair e o programa ser finalizado.

3. 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  ->  boolean ;; ...
```

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)
```