

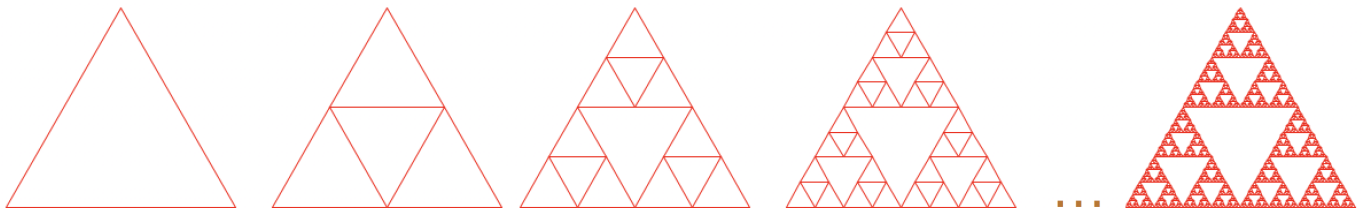
Lista de Exercícios 11 (Caps. 25 e 26) – INF05008

Data limite para submissão: 9 de novembro às 23:59

- USE OS NOMES DOS TIPOS DE DADOS E FUNÇÕES DEFINIDOS NAS QUESTÕES (a grafia deve ser exatamente a indicada nas questões).
- Use o **template da solução disponível no Moodle**, onde há algumas definições de funções e do tipo **Figura**, que é usado em várias questões.
- Não precisa colocar testes nas funções desta lista que produzem imagens. Deixe apenas as chamadas para que quando seu arquivo for rodado, as imagens sejam geradas.
- Para todas as funções recursivas definidas por vocês, mostrem o modelo de solução (a seguir) e argumentem sobre a terminação da função. Lembrem que o problema pode ter vários casos triviais e vários casos mais complexos.

```
;; MODELO DE SOLUÇÃO:
;; Se <problema trivial> então <solução do problema trivial>
;; Se <problema complexo> então
;;     <combinar soluções>
;;     <solucionar subproblema 1> <gerar subproblema 1>
;;     ...
;;     <solucionar subproblema n> <gerar subproblema n>
```

1. Complete o código da função **sierpinski** que, dados o tamanho do lado e uma cor, desenha um triângulo de Sierpinski desta cor cujo lado do triângulo externo é o lado passado como argumento. Se o lado passado como argumento tiver dimensões muito pequenas (você escolhe o que é "*dimensão muito pequena*"), somente desenha um triângulo. O triângulo de Sierpinski é construído dividindo-se cada triângulo em 3, e desenhando recursivamente um triângulo dentro de cada um destes triângulos, conforme os desenhos a seguir:



2. Construa a função **desenha-sierpinski** para gerar um triângulo de Sierpinski recebendo um argumento do tipo **Figura**. A função deve desenha um triângulo de Sierpinski equilátero com as especificações dadas na figura passada como argumento. Utilize as funções **gera-cor** e **sierpinski** para definir a cor da figura e desenha o triângulo de Sierpinski.

Obs: Lembre que em um triângulo equilátero, a relação entre a altura h e o lado l é dada por $h = \frac{l\sqrt{3}}{2}$.

3. Construa a função **desenha-boneco** que, dado um argumento do tipo **Figura**, desenha um boneco palito nesta cor, e com a altura dada. A altura do boneco deve ser a distância entre o topo de sua cabeça e o chão. Uma sugestão para definir os tamanhos das partes é dividir esta distância em um número de unidades e utilizar esta unidade para desenha cada parte. Para construir o boneco, pode-se definir um **local** com o grupo de partes do boneco, que depois serão usadas para montar o boneco (veja o código parcialmente preenchido abaixo). Para desenha o corpo, braço e pernas pode-se usar a função **line** que, dadas 2 coordenadas x e y e uma cor, desenha uma linha a partir do ponto superior esquerdo (ponto (0,0)) até este ponto. Pode-se usar outras funções pré-definidas de desenho, veja o manual do pacote **image.rkt** em <https://docs.racket-lang.org/teachpack/2htdpimage.html>:

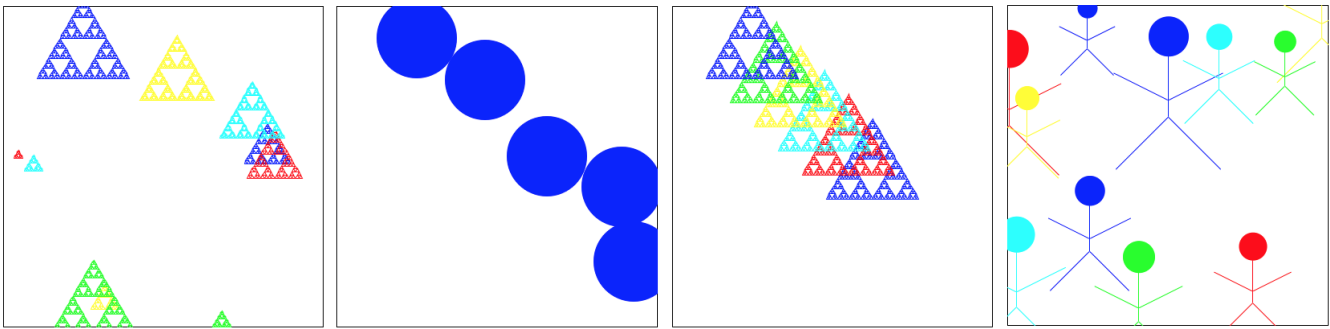
```
;; desenha-boneco: Figura -> Imagem
(define (desenha-boneco fig)
  (local
    (
      (define TAM (figura-altura fig))
      (define COR (gera-cor (figura-cor fig)))
      (define CABECA ...)
```

```

(define CORPO ... )
(define BRACO-DIR ... )
(define BRACO-ESQ ...)
(define PERNA-DIR ... )
(define PERNA-ESQ ...)
)
(above
  CABECA
  (beside BRACO-DIR CORPO BRACO-ESQ)
  (beside PERNA-DIR PERNA-ESQ))))

```

4. Faça um programa chamado **desenha-figuras** que desenha várias figuras em uma cena, variando pelo menos um dos seus atributos (cor, localização, altura). O programa recebe a função que desenha a figura e uma figura (tipo **Figura**) contendo a posição inicial, o tamanho inicial e um número (representando a cor inicial). O programa pode variar a cor, a posição da figura e/ou o seu tamanho, até que um critério de fim seja encontrado. Defina como quiser a próxima figura a ser desenhada (como muda a cor, tamanho, posição) e qual o critério de fim. Alguns exemplos podem ser vistos na figura abaixo (nestas figuras foi usada uma cena com largura e altura de 400 pontos).



5. Generalize a função do exercício anterior:

- (a) Construa funções para testar critérios diferentes de fim e diferentes opções de alteração de figuras (pelo menos 3 funções de critérios e 3 funções para alterar figuras). Pelo menos uma das funções de alteração deve variar também a cor.
- (b) Construa a função de **desenha-figuras-gen**, que recebe uma função de desenhar figura, uma figura inicial, um critério de fim e uma função de alteração de figuras e gera uma cena com várias figuras (como as da questão anterior). Use as funções do item a) para gerar exemplos diferentes para a função **desenha-figuras-gen**. Construa pelo menos 4 chamadas diferentes da sua função. A seguir, exemplos de chamadas que geraram as figuras da questão anterior – note que, com excessão dos elementos (**make-figura...**) todos outros argumentos da função **desenha-figuras-gen** são funções, que devem ser definidas para poder executar essas chamadas:

```
(desenha-figuras-gen desenha-sierpinski (make-figura 100 40 100 1) pequena? diminui-alt)
```

```
(desenha-figuras-gen desenha-bola (make-figura 100 40 100 1) out-of-borders? move-dir-random)
```

```
(desenha-figuras-gen desenha-sierpinski (make-figura 100 40 100 1) cor>7? move-dir)
```

```
(desenha-figuras-gen desenha-boneco (make-figura 100 40 100 1) grande? aumenta-alt)
```

Para gerar um movimento aleatório, pode-se usar a função **random**, que recebe um número natural e gera um número natural entre 0 e este número como resultado.

- (c) Como a função **desenha-figuras-gen** é muito genérica, e seu critério de fim e de avanço (movimentação da figura) são argumentos da função, não é possível fazer uma argumentação genérica sobre a terminação desta função: dependendo da chamada, ela pode terminar ou não. Argumente sobre a terminação das 4 chamadas que você realizou na questão anterior.

Desafio (não vale nota): Você conseguiria gerar o "tapete de Sierpinski"? E tapetes coloridos (neste caso, pode-se usar um argumento a mais como a cor inicial, ou gerar esta cor randomicamente)?

