Exercícios Extra-classe – Laboratório 4

Atenção:

- **Não esqueça que a documentação faz parte das soluções!**
- **Este trabalho pode ser realizado em duplas.**
- **Denomine seu arquivo a ser submetido via Moodle da seguinte forma: `listalab4C_<seus nomes aqui>.rkt`;**
- **Todas as funções devem ser testadas com `check-expect` (mínimo 2 testes por função), a exceção são as funções que envolvem algum componente randômico na resposta, deixe apenas indicados os testes e os resultados esperados**
- **Arquivos que não estejam de acordo com estas regras ou que não compilarem não serão corrigidos**

O objetivo do programa `amigo-secreto` abaixo é gerar as atribuições de nomes para um amigo-secreto: o primeiro nome da lista de entrada pegaria a primeira pessoa da lista de saída, e assim por diante.

```
;; amigo-secreto: Lista-de-nomes -> Lista-de-nomes
;; obj: Dada uma lista de nomes, gerar randomicamente uma lista
;;      de nomes na qual cada nome da lista de saída está em uma
;;      posição diferente do que na lista de entrada.
 (define (amigo-secreto lista)
       (selecao-random
                (nomes-diferentes lista (arranjos lista))))
```

1. Defina os tipos de dados `Lista-de-nomes` e `Lista-de-lista-de-nomes` para representar, respectivamente, lista de nomes de pessoas e listas de listas de nomes de pessoas.
2. Desenvolva as funções auxiliares:
   (a) `selecao-random : Lista-de-lista-de-nomes -> Lista-de-nomes`
   Dada uma lista de listas de nomes, seleciona randomicamente um de seus elementos. (Pode-se usar a função `random:Número->Número` , que dado um número n, devolve um número entre 0 e $n-1$.)
   (b) `nomes-diferentes: Lista-de-nomes Lista-de-lista-de-nomes -> Lista-de-lista-de-nome`
   Dada uma lista de nomes L e uma lista com arranjos de L, devolve a lista de arranjos de L nos quais cada nome não está na mesma posição na lista L e no arranjo.
   (c) `arranjos: Lista-de-nomes -> Lista-de-lista-de-nomes`
   Dada uma lista de nomes L, devolve todos os possíveis arranjos com os elementos de L. Por exemplo:

```
(arranjos (list 'Ana 'Pedro 'Paulo)  produz como resultado
          (list (list 'Ana 'Pedro 'Paulo)
                (list 'Ana 'Paulo 'Pedro)
                (list 'Paulo 'Ana "Pedro)
                (list 'Paulo "Pedro 'Ana)
                (list 'Pedro 'Ana "Paulo)
                (list 'Pedro 'Paulo 'Ana))
```

   (A função desenvolvida por você não precisa devolver os elementos da lista necessariamente nesta ordem, mas devem ser estes os elementos da lista resultante.)

Obs.: Provavelmente várias outras funções auxiliares serão necessárias. Lembrem: sempre que o problema se tornar muito difícil, pensem em utilizar uma função auxiliar que resolve parte do problema. A função `arranjos` é um desafio de programação, nesta questão dividir o problema em problemas menores é essencial!