Lista de Linguagens de Programação – 6

Nome: _____ Matrícula: _____

1. Escreva uma função `member`, de tipo `''a * ''a list -> bool`, de modo que `member(e, L)` retorne verdade se, e somente se, `e` for um elemento da lista `L`.

2. Escreva uma função `less`, de tipo `int * int list -> int list`, tal que `less(e, L)` retorne a lista formada por todos os inteiros de `L` que são menores que `e`.

3. Escreva uma função `repeats`, de tipo `''a list -> bool` tal que `repeats(L)` seja verdade se, e somente se, a lista `L` possui dois elementos consecutivos e iguais.

4. Neste exercício, um polinômio será representado usando uma lista com os seus coeficientes reais, começando com a constante, e tendo como último elemento o coeficiente de grau mais alto. Por exemplo, $3x^2 + 5x + 1$ seria representado como a lista `[1.0, 5.0, 3.0]`. Já $x^3 - 2x$ seria representado como `[0.0, 2.0, 0.0, 1.0]`. Escreva uma função `eval`, de tipo `real list * real -> real` que receba um polinômio representado desta forma, mais um valor de x, e retorne o valor do polinômio para aquele dado x. Por exemplo, `eval([1.0, 5.0, 3.0], 2.0)` deveria produzir o resultado `23.0`, já que, se $x = 2$, então $3x^2 + 5x + 1 = 23$.

5. Escreva uma função `quicksort`, de tipo `int list -> int list`. Caso você tenha esquecido, o quicksort funciona assim: primeiro pegue um elemento da lista. Este elemento será chamado *pivot*. Particione o resto da lista em duas sub-listas, uma com elementos menores que o pivot, e a outra com elementos maiores ou iguais. recursivamente ordene as sub-listas. Combine as duas sub-listas, mais o pivot, para formar a lista final.

6. Funções podem ser passadas como parâmetros, da mesma forma que qualquer outro valor da linguagem ML. Por exemplo, considere as definições de funções abaixo:

```
fun square a = a * a;
fun double a = a + a;
fun compute (n, f) = f n;
```

As funções `square` e `double` recebem um único inteiro como parâmetro e retornam um inteiro como resultado. A função `compute` recebe dois parâmetros, um valor `n` e uma outra função `f`, e retorna o resultado da aplicação da função ao parâmetro. Assim, `compute(3, square)` produz o resultado `9`, enquanto `compute(3, double)` produz `6`. Nós ainda vamos falar muito destas *funções de alta ordem*. Mas neste exercício a gente já começa a ver um pouquinho disto.

Escreva uma outra versão da função `quicksort`, desta vez de tipo `'a list * ('a * 'a -> bool) -> 'a list`. O segundo parâmetro é uma função que faz o papel de *comparador*.

E por que definir um quicksort assim? Porque esta nova função é muito mais útil e mais reutilizável que a função original! Por exemplo, suponha que tenhamos definido duas funções de comparação assim:

```
fun icmp (a, b) = a < b;
fun rcmp (a: real, b) = a < b;
```

Nós poderiamos usar o novo quicksort via uma chamada assim `quicksort(L, icmp)` para ordenar uma lista de inteiros em ordem crescente, mas poderíamos também usar `quicksort(M, rcmp)` para ordenar uma lista `M` de reais. Além disto, se você definisse:

```
fun ircmp (a, b) = a > b;
```

Então poderíamos usar `quicksort(L, ircmp)` para ordenar uma lista `L` de inteiros em ordem decrescente!

7. Nos próximos exercícios vamos representar conjuntos como listas. Cada elemento da lista aparece uma única vez, mas não existe nenhuma ordem em particular. Não assuma que as listas estão ordenadas, mas assuma que as listas passadas como entradas não possuem elementos repetidos. Em suas respostas, é preciso garantir que as listas de saída não terão elementos repetidos.

   (a) Escreva uma função `isIn`, que testa se um determinado elemento passado como parâmetro é membro de um conjunto.

   (b) Escreva uma função para produzir a união de dois conjuntos.

   (c) Escreva uma função `powerset`, de tipo `'a list -> 'a list list` que construa o conjunto potência de um conjunto. O conjunto potência de C é o conjunto de todos os subconjuntos de C. Por exemplo, o conjunto potência de $A = \{1, 2, 3\}$ é: $\{x | x \subseteq A\} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$. Sua função `powerset` deve receber uma lista, representando o conjunto de entrada, e retornar uma lista de listas, em qualquer ordem, representando todos os possíveis subconjuntos. A sua função não precisa funcionar em uma lista vazia não tipada. Isto é, um erro pode acontecer quando você receber `nil`. Por outro lado, sua função precisa funcionar para `(nil: int list)`. Neste caso, o resultado seria `([[]])`.