Lista de Linguagens de Programação – 5

Nome: _____ Matrícula: _____

1. Escreva uma função `cube`, de tipo `int -> int` que retorne o cubo do seu parâmetro.

2. Escreva uma função `cuber`, de tipo `real -> real` que retorne o cubo de seu parâmetro.

3. Escreva a função `fourth`, de tipo `'a list -> 'a` que retorne o quarto elemento da lista passada como parâmetro. Não se preocupe com a condição de contorno em que a lista de entrada possui menos que quatro elementos.

4. Escreva uma função `min3`, de tipo `int * int * int -> int` que retorne o menor elemento de uma tupla de três inteiros.

5. Escreva uma função `red3`, de tipo `'a * 'b * 'c -> 'a * 'c` que converte uma tupla com três elementos em uma tupla com dois elementos, eliminando o segundo elemento da entrada.

6. Escreva uma função `thirds`, de tipo `string -> char` que retorne o terceiro caracter de uma *string*. Não se preocupe com *strings* com menos que três letras.

7. Escreva uma função `cycle1`, de tipo `'a list -> 'a list`, que receba uma lista l_i de entrada e retorne uma lista l_o de saída tal que o primeiro elemento de l_i tenha sido movido para a última posição de l_o. Por exemplo, `cycle1 [1, 2, 3, 4]` deve retornar `[2, 3, 4, 1]`.

8. Escreva uma função `sort3`, de tipo `real * real * real -> real list` que retorne uma lista ordenada com os três números passados como entrada.

9. Escreva uma função `del3`, de tipo `'a list -> 'a list`, cuja lista de saída seja a mesma que a lista de entrada, mas com o terceiro elemento removido. Novamente, a sua função não precisa tratar listas com menos que três elementos.

10. Escreva uma função `sqsum`, de tipo `int -> int` que receba um número não negativo n e retorne a soma de todos os quadrados de 0 até n. Sua função não precisa se comportar bem com entradas menores que 0.

11. Escreva uma função `cycle`, de tipo `'a list * int -> 'a list` que receba uma lista e um número inteiro e retorne a mesma lista, mas com os n primeiros elementos movidos para o final da lista. Por exemplo, `cycle ([1, 2, 3, 4, 5, 6], 2)` deve retornar `[3, 4, 5, 6, 1, 2]`.

12. Escreva uma função `pow`, de tipo `real * int -> real` que eleve um número real a uma potência inteira. Não é preciso tratar o caso em que a potência é negativa.

13. Escreva uma função `max`, de tipo `int list -> int` que retorne o maior elemento de uma lista de inteiros. Não é preciso tratar o caso em que a lista é vazia. *Dica:* escreva uma função auxiliar `maxhelper` que receba como segundo parâmetro o maior elemento visto até o momento. Depois, é só fazer:

    ```
    fun max x = maxhelper (tl x, hd x)
    ```

14. Escreva uma função `isPrime`, de tipo `int -> bool` que retorne verdadeiro se o parâmetro é um número inteiro primo. Não é preciso tratar o caso em que a entrada é negativa.

15. Escreva uma função `select`, de tipo `'a list * ('a -> bool) -> 'a list`, que receba uma lista e uma função f como parâmetros. `select` deve, então, aplicar f a cada elemento da lista de entrada, de modo a retornar uma nova lista contendo somente aqueles elementos e para os quais $f(e)$ é verdade. Os elementos da saída podem ser enfileirados em qualquer ordem. Por exemplo: `select ([1, 2, 3, 4, 5, 6, 7, 8, 9, 10], isPrime)` retornará uma lista como `[7, 5, 3, 2]`. Este é um exemplo de uma *função de alta ordem*, pois `select` recebe outra função como parâmetro de entrada. Nós veremos mais sobre isto nas próximas aulas.