ECOM11 - Haskell – Lista de Exercícios

UNIFEI – Universidade Federal de Itajubá
Prof. João Paulo R. R. Leite (joaopaulo@unifei.edu.br)

Data para Entrega: 06/10/2020 até as 23:59h.

Resolva os seguintes exercícios utilizando a linguagem **Haskell**, e submeta o arquivo com as soluções na tarefa cadastrada no SIGAA chamada "Trabalho de Implementação – Haskell". As soluções para os exercícios devem estar todos em um mesmo arquivo, com extensão ".hs", organizados através de comentários (para que eu saiba qual solução é de qual exercício). No arquivo, como comentário, você deve escrever seu nome completo e seu número de matrícula.

O trabalho valerá 35% da primeira nota (N1).

Lista de Exercícios:

1. Escreva uma função para calcular a **área** de um retângulo de lados w e h, e outra para calcular a **área** de um triângulo com base b e altura h.

2. Escreva uma função que calcule o **tempo gasto** por um carro de corrida para percorrer uma dada distância s, com velocidade média igual a v. Depois disso, escreva uma função para calcular a diferença de tempo entre dois carros após percorrerem uma mesma distância s, mas com velocidades diferentes v1 e v2.

3. Utilizando **guardas**, escreva uma função em Haskell para calcular a seguinte função **recursiva**:

$$L(n) = \begin{cases} 1 & \text{se } n = 0 \\ 3 & \text{se } n = 1 \text{ ou } n = 2 \\ L(n-1) + L(n-2) & \text{se } n > 2. \end{cases}$$

4. Utilizando **guardas**, escreva uma função em Haskell para calcular o valor da "Função de Ackerman", de acordo com a definição abaixo. A função de Ackerman retorna valores absurdamente grandes, portanto seja cauteloso. Faça apenas os seguintes testes: A(0,10), A(4,0), A(3,2). As respostas serão, respectivamente, 11, 13 e 29.

$$A(m,n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1,1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m-1, A(m,n-1)) & \text{se } m > 0 \text{ e } n > 0. \end{cases}$$

5. Utilizando **guardas**, escreva uma função que receba o valor das arestas de um triângulo e indique se o triângulo é escaleno, isósceles ou equilátero. A função deve retornar um valor inteiro: 1 para escaleno, 2 para isósceles e 3 para equilátero.

6. Escreva uma função que receba dois números reais. A função deve retornar -1 se o primeiro parâmetro for menor que o segundo, 0 se os dois parâmetros forem iguais e a multiplicação a*b se o primeiro parâmetro for maior que o segundo. Utilize **guardas**.

7. Escreva uma função em Haskell que execute um processamento semelhante ao descrito pela função a seguir, escrita na l**inguagem C**:

```
void soma (int N, int *R) {
  int i;
  *R = 0;
  for (i=N; i>0; i--)
    if (i % 2 == 1)
       *R += i;
}
```

8. Escreva uma função **recursiva** que verifique se um determinado número é **primo** ou não. A função deve retornar um valor booleano (True ou False).

9. Escreva uma função utilizando a **recursão em cauda** para calcular o enésimo número da sequência de **Fibonacci**. Lembre-se de que cada número da sequência de Fibonacci é a soma dos dois anteriores: 1, 1, 2, 3, 5, 8, 13, 21, etc.

10. Vamos relembrar algumas fórmulas do **movimento uniformemente variado**:

$$v = v_0 + at$$
$$\Delta s = v_0\, t + \frac{a}{2} t^2$$

    Escreva uma função que receba a velocidade inicial e final de um corpo, e o tempo gasto para ir de uma a outra, e retorne a distância percorrida pelo corpo durante a aceleração. Faça uma versão utilizando definições locais com **where** e outra com **let/in**.

11. Escreva uma função recursiva para calcular a **soma de dois números naturais** através do seguinte método: repetidamente acrescente cada unidade do número menor ao número maior, até que o número menor seja igual a zero.

12. Escreva uma função que receba duas listas de inteiros ordenadas, sem elementos repetidos, e retorne uma **lista com os elementos que só fazem parte de uma única lista**, ou seja, caso um número faça parte das duas listas, ele não estará presente no resultado da função. A lista resultante também deve estar ordenada.

13. Escreva uma função que retorne o **enésimo elemento** de uma lista de inteiros. A função deve receber o inteiro n e a lista de inteiros, e retornar o elemento inteiro na posição n.

14. Escreva uma função para **inserir um elemento na enésima posição** de uma lista de inteiros passada como parâmetro. A posição n também deve ser passada como parâmetro. A função retorna a lista contendo o novo elemento na posição adequada.

15. Escreva uma função para **substituir todas as ocorrências** de um determinado caractere por outro em uma string passada como parâmetro. Os caracteres também devem ser passados como parâmetro. A função retorna a nova string, obtida após a substituição.

16. Faça uma função para verificar se uma string passada como parâmetro é um **palíndromo** (mesma nos dois sentidos). A palavra deve ser representada por uma lista de caracteres. A função deve retornar um valor booleano.

17. Defina duas funções, chamadas "**iniciais**" e "**finais**", que recebem, cada uma, uma lista de inteiros e um inteiro n, e retornam outras listas, com os n primeiros e n últimos elementos da lista original.

18. Dadas duas listas de inteiros, escreva uma função chamada "**prefixo**" que retorne True se a primeira lista for um prefixo da segunda, e False caso contrário. A lista X é um prefixo da lista Y se todos os elementos de X aparecem em Y antes de quaisquer outros elementos da Y, na mesma ordem.

19. Dado uma lista de listas de inteiros, escreva uma função chamada "**sozinhos**" que retorne a lista composta pelos elementos que aparecem em listas sozinhas na lista de listas. A lista sozinha é uma lista com exatamente um elemento.

20. Dada uma lista de inteiros, escreva uma função chamada "**positivos**" que remova desta lista todos os inteiros menores do que zero e retorne a lista filtrada.

Bom trabalho!