Disciplina: Paradigmas de Programação
Professor: Maicon Rafael Zatelli
Entrega: *Moodle*

Atividade V - Haskell

Atenção: Faça um ZIP com todos os arquivos de solução. Use o nome do arquivo de maneira a entender qual problema você está resolvendo. Por exemplo, problema1.hs, problema2.hs e assim por diante.

Resolva os seguintes problemas na linguagem Haskell:

TENTE RESOLVER OS PROBLEMAS SEM UTILIZAR **map**, **filter** OU QUAISQUER OUTRAS FUNÇÕES JÁ PRONTAS DO HASKELL, A MENOS QUE EXPLICITAMENTE REQUISITADO NO PROBLEMA.

1. Crie uma função com assinatura `soma :: [Int] -> Int`, a qual recebe uma lista de Int e retorna a soma de todos os elementos da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa.
2. Crie uma função com assinatura `media :: [Int] -> Float`, a qual recebe uma lista de Int e retorna a média de todos os elementos da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa. DICA: utilize a função `fromIntegral` para converter um tipo inteiro para um tipo compatível com o operador de divisão `/`
3. Crie uma função com assinatura `menor :: [Int] -> Int`, a qual recebe uma lista de Int e retorna o menor elemento da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa.
4. Crie uma função com assinatura `diferencaMaiorMenor :: [Int] -> Int`, a qual recebe uma lista de Int e retorna a diferença entre o maior e o menor elemento da lista. Retorne 0 caso a lista for vazia. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa.
5. Crie uma função com assinatura `busca :: [Int] -> Int -> Bool`, a qual recebe uma lista de Int e um Int e retorna se o elemento passado como parâmetro encontra-se na lista ou não. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa.
6. Crie uma função com assinatura `ocorrencias :: [Int] -> Int -> Int`, a qual recebe uma lista de Int e um Int e retorna o número de vezes em que o elemento está presente na lista. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa.
7. Motifique o arquivo `alunos.hs` (disponível no Moodle) de forma a adicionar novas funções:
   - **A:** Crie uma função com a seguinte assinatura: `aprovados :: [(Int, String, Float)] -> [String]`, a qual recebe uma lista de alunos e retorna uma lista com o nome dos alunos aprovados. Um aluno está aprovado se a sua média é maior ou igual a 6. Utilize **map** e **filter** para resolver esta questão.
   - **B:** Crie uma função com a seguinte assinatura: `aprovados2 :: [(Int, String, Float)] -> [String]`, a qual recebe uma lista de alunos e retorna uma lista com o nome dos alunos aprovados. Um aluno está aprovado se a sua média é maior ou igual a 6. Não utilize **map** e **filter** para resolver esta questão. **Utilize** o conceito de **list comprehension**.
   - **C:** Utilize (e modifique, se necessário) a função `gerarPares` vista em aula e disponível no arquivo `alunos.hs` para formar duplas de alunos. Note que um aluno não pode fazer dupla consigo mesmo.
8. Crie uma função com assinatura `inverte :: [t] -> [t]`, a qual recebe uma lista como parâmetro e deve retornar a mesma invertida. **Não utilize** nenhuma função pronta do Haskell para realizar esta tarefa.
9. Crie uma função com assinatura `mapear :: (t -> y) -> [t] -> [y]`, a qual recebe uma função, uma lista e retorna uma lista. Esta função `mapear` fará o mesmo que a função `map`, ou seja, aplicar a função recebida como parâmetro sobre cada elemento da lista e retornar a lista resultante. **Não utilize map** ou **filter** para esta tarefa.

10. Crie uma função com assinatura `filtrar :: (t -> Bool) -> [t] -> [t]`, a qual recebe uma função, uma lista e retorna uma nova lista. Esta função aplica a função recebida como parâmetro sobre cada elemento da lista e caso o retorno da função for verdadeiro, então o elemento fará parte da nova lista, caso contrário não. Para esta tarefa, **utilize** o conceito de **list comprehension**.

11. Crie uma função com assinatura `primeiros :: Int -> [t] -> [t]`, a qual recebe um número de elementos, uma lista, e retorna uma lista. Esta função deve retornar uma lista com os n primeiros elementos informados no primeiro parâmetro. **Não utilize** nenhuma função pronta to Haskell para esta tarefa.

12. Crie uma função com assinatura `apagar :: Int -> [t] -> [t]`, a qual recebe um número de elementos, uma lista, e retorna uma lista. Esta função deve remover da lista os n primeiros elementos fornecidos como parâmetro. Por exemplo, a chamada `(apagar 3 [1,2,3,4,5])` deve retornar `[4,5]`. **Não utilize** nenhuma função pronta to Haskell para esta tarefa.

13. Crie uma função com assinatura `apagarEnquanto :: (t -> Bool) -> [t] -> [t]`, a qual recebe uma função como parâmetro e uma lista, e retorna uma lista. Esta função deve aplicar a função passada como parâmetro a cada elemento da lista, até que algum elemento da lista retorne `False` na aplicação da função. Os elementos da lista resultante são então todos os elementos a partir do elemento em que a função passada como parâmetro retornou `False`. Por exemplo a chamada `(apagarEnquanto par [2,4,1,2,3,4,5])` deve retornar `[1,2,3,4,5]`, visto que ao testar o elemento 1, a função `par` retorna `False`. **Não utilize** nenhuma função pronta to Haskell para esta tarefa.