Princípios de Programação

Trabalho Primeiro (v2)

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2020/2021

Neste trabalho primeiro vamos desenvolver três pequenas aplicações.

A. Detecção de Plágio Uma das técnicas envolvidas na detecção de plágio passa por analisar a frequência de letras, palavras ou expressões num texto. Pretende-se implementar uma função `frequencias` que converte um texto (uma `String`) na lista de frequências com que cada caracter aparece nesse mesmo texto.
Apresentamos de seguida alguns exemplos da sua utilização.

```
> frequencias "ABBa"
[1,2,2,1]
> frequencias "ATDI - napoleon solo"
[1,1,1,1,3,1,3,2,1,1,4,2,1,4,2,3,1,4,2,4]
> frequencias "Taylor Swift"
[1,1,1,1,1,1,1,1,1,1,1]
> frequencias ""
[]
```

B. Pequenas Palavras Para além das línguas naturais, existem também as línguas artificiais como o Esperanto, Dothraki, Sindarin ou Klingon. Neste exercício vamos fazer um pequeno estudo que nos permita desenhar uma língua que seja mais concisa e eficaz. Para tal, gostaríamos que os conceitos mais usados sejam representados por palavras de três letras ASCII apenas. No entanto, essas palavras têm de ser pronunciáveis, bastando para isso conterem pelo menos uma vogal (a, e, i, o, u, ou y).
Defina a expressão `pequenasPalavras` que devolva, por ordem lexicográfica, todas as palavras que cumpram estes requisitos. De seguida apresentam-se alguns usos desta expressão.

```
> take 10 pequenasPalavras
["aaa","aab","aac","aad","aae","aaf","aag","aah","aai","aaj"]
> length pequenasPalavras
9576
> pequenasPalavras !! 1459
"dvy"
> pequenasPalavras !! 1460
"dwa"
```

C. Campainhas de Prédios No Japão e em outros países asiáticos, não existe o quarto andar em prédios. Por motivos similares, em algumas culturas ocidentais não existe o 13º andar.
Pretende-se implementar o software que mostra numa campainha digital os vários andares, de forma a que os visitantes, correios, transportadoras ou agentes comerciais inoportunos possam escolher qual dos apartamentos pretendem chamar.
Para tal, deverá implementar a função `legendaCampainha` que irá devolver a lista ordenada com as várias legendas a colocar no menu. De forma a que esta função se adapte a qualquer prédio e cultura, deverá receber o número total de andares existente no prédio, o número a evitar nessa cultura, e uma lista de pares indicando a posição dentro do andar ("Dto"ou "Esq", "A", "B"ou "C", etc.) e quantos andares têm essa posição, contando de baixo.
Esta última restrição permite ter prédios com diferentes topologias. Por exemplo, é frequente o último andar ter apenas um apartamento de dimensões superiores.
Nos prédios que vão usar o nosso sistema, sabemos que o rés-do-chão contém apenas lojas, pelo que não necessitamos de o considerar. Os andares subterrâneos são reservados para garagens, pelo que também não serão considerados.
Poderá ainda considerar que

- O número total de andares é não negativo.
- O número a evitar é também ele não negativo. Poderá, no entanto, ser superior ao número total de andares do prédio em questão.
- A quantidade de andares com uma certa tipologia nunca será negativa e nunca será superior ao número total de andares.

Para ajudar à compreensão, pedimos ao arquitecto responsável que nos fornecesse alguns exemplos.
Exemplo 1: Um prédio com 10 andares no Japão (evitar o número 4) com Direito em todos os andares, Esquerdo só até ao oitavo e Central até ao terceiro. De notar que, como saltamos o quarto, os andares na legenda vão até ao 11º, apesar de apenas existirem 10.

```
> legendaCampainha 10 4 [("Dto", 10), ("Esq", 8), ("Cent", 3)]
["1Dto","1Esq","1Cent","2Dto","2Esq","2Cent","3Dto","3Esq",
"3Cent","5Dto","5Esq","6Dto","6Esq","7Dto","7Esq","8Dto",
"8Esq","9Dto","9Esq","10Dto","11Dto"]
```

Exemplo 2: Neste prédio ocidental (evitar o 13º andar) de três andares, temos o primeiro andar com A, B, C e D, o segundo andar com A e B e o último andar apenas com A.

```
> legendaCampainha 3 13 [("A", 3), ("B", 2), ("C", 1), ("D", 1)]
["1A","1B","1C","1D","2A","2B","3A"]
```

Exemplo 3: Este prédio é composto apenas por lojas, pelo que o total de andares residenciais é 0. As tipologias inicialmente previstas (Norte e Oeste) não existem em nenhum andar.

```
> legendaCampainha 0 4 [("Norte", 0), ("Oeste", 0)]
[]
```

Notas

1. Os trabalhos serão avaliados automaticamente. Respeite os nomes e os tipos de todas funções enunciadas acima.
2. Cada função (ou expressão) que escrever deverá vir sempre acompanhada de uma assinatura. Isto é válido para as funções enunciadas acima bem como para outras funções ajudantes que decidir implementar.
3. Para resolver estes problemas deve utilizar *apenas* a matéria dos três primeiros capítulos do livro (de "Starting Out" até "Syntax in Functions"). Pode usar qualquer função constante no `Prelude`. Não pode em caso algum utilizar funções definidas por recursão.
4. Lembre-se que as boas práticas de programação Haskell apontam para a utilização de várias funções simples em lugar de uma função única mas complicada.

Entrega. Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 5 de outubro de 2020.

Plágio. Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Relembramos aqui um excerto da sinopse: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".