Trabalho Prático 1

Enunciado

Objetivo: Implementação de uma biblioteca de big-numbers em Haskell, incluindo operações aritméticas básicas, para cálculo da sequência de Fibonacci.

Nota: Cada função descrita neste trabalho prático deverá ser definida com o nome indicado (a verde). Os ficheiros com o código das funções principais deverá ter os nomes indicados (a amarelo). O não cumprimento destas indicações será penalizado. Se necessário, poderá desenvolver funções e/ou módulos auxiliares.

1. Crie um ficheiro chamado Fib.hs. Nesse ficheiro, implemente três versões do cálculo do enésimo número de Fibonacci. Considere que os índices da sequência de Fibonacci começam em 0: fib(0) = 0, fib(1) = 1, fib(2) = fib(0) + fib(1) = 0 + 1 = 1, …
   1.1) Uma função recursiva, fibRec;
   O tipo desta função é: `fibRec :: (Integral a) => a -> a`
   1.2) Crie uma versão otimizada da função anterior, chamada fibLista, usando uma lista de resultados parciais tal que (`lista !! i`) contém o número de Fibonacci de ordem i (programação dinâmica).;
   1.3) Implemente outra versão, chamada fibListaInfinita, para gerar uma lista infinita com todos os números de Fibonacci e retornar o elemento de ordem n.

2. [parte mais valorizada do trabalho] Implemente um módulo de aritmética para big-numbers e utilize-o para implementar novas versões das funções definidas na pergunta 1 para este novo tipo. Este módulo deverá ser definido num ficheiro chamado BigNumber.hs.

   Um big-number é representado por uma lista dos seus dígitos. O objetivo deste módulo é representar estruturas para big-numbers (por exemplo listas) e implementar as funções básicas de aritmética para big-numbers. Os big-numbers podem ser positivos ou negativos.

   O módulo deverá conter as seguintes definições:

   2.1) Uma definição do tipo BigNumber.

   2.2) A função scanner, que converte uma string em big-number. O seu tipo é: `String -> BigNumber`

   2.3) A função output, que converte um big-number em string. O seu tipo é: `BigNumber -> String`

   2.4) A função somaBN, para somar dois big-numbers.

   2.5) A função subBN, para subtrair dois big-numbers.

   2.6) A função mulBN, para multiplicar dois big-numbers.
   O tipo das três funções anteriores é: `BigNumber -> BigNumber -> BigNumber`

   2.7) A função divBN, para efetuar a divisão inteira de dois big-numbers. A divisão deverá retornar um par "(quociente, resto)". Assuma que ambos os argumentos são positivos.
   O tipo desta função é: `divBN :: BigNumber -> BigNumber -> (BigNumber, BigNumber)`

   Para facilitar estas operações, assuma que os dígitos podem estar na lista por ordem inversa, chamando inicialmente uma função de reverse. Exemplo:

   123 + 49 = 172 é representado por:

   `[3, 2, 1] + [9, 4] = [2, 7, 1]`

3. No ficheiro Fib.hs, inclua o módulo dos big-numbers e implemente uma versão das três funções da alínea 1 que trabalhe com big-numbers. As três funções deverão chamar-se (respectivamente) fibRecBN, fibListaBN, fibListaInfinitaBN.

4. Compare as resoluções das alíneas 1 e 3 com tipos (`Int -> Int`), (`Integer -> Integer`) e (`BigNumber -> BigNumber`), comparando a sua aplicação a números grandes e verificando qual o maior número que cada uma aceita como argumento.

5. Acrescente ao módulo de big-numbers da alínea 2 a capacidade de detetar divisões por zero em *compile-time*. Para isso, a função divisão deverá retornar *monads* do tipo `Maybe`. A função alternativa para a divisão inteira deverá se chamar safeDivBN e ter o tipo:
   `safeDivBN :: BigNumber -> BigNumber -> Maybe (BigNumber, BigNumber)`
   Mais informações sobre o *monad Maybe* em:
   https://en.wikibooks.org/wiki/Haskell/Understanding_monads/Maybe

Condições de Realização

Constituição dos Grupos: grupos de **2 estudantes**, inscritos na mesma turma teórico-prática. Excecionalmente e apenas em caso de necessidade, podem aceitar-se grupos de 3 elementos.

Escolha de grupos: os grupos deverão ser indicados na atividade a disponibilizar para o efeito a partir do dia **8 de Novembro** de 2021, no *Moodle*. Numa primeira fase, um dos elementos do grupo deverá fazer a escolha do grupo (um qualquer grupo da turma prática que frequentem); numa segunda fase, o segundo elemento juntar-se-á ao grupo.

Prazos: a escolha de grupos deverá ser realizada durante as próximas duas semanas. A entrega do trabalho (código-fonte e ficheiro readme) deverá ser realizada até ao final do dia **28 de Novembro** de 2021, na atividade a disponibilizar para o efeito no Moodle, e a demonstração do trabalho deve ser realizada na semana de 29 de Novembro de 2021.

Pesos das Avaliações: ver ficha da Unidade Curricular no SIGARRA.

Linguagens e Ferramentas: o trabalho deve ser desenvolvido em linguagem Haskell, devendo ser garantido o seu funcionamento em Windows e Linux, usando o ghci, versão 9.0.1. Caso seja necessária alguma configuração (para além da instalação padrão do *software*), isso deverá estar expresso no ficheiro README que deverá ainda incluir os passos necessário para configurar e/ou instalar as componentes necessárias (em Windows e Linux). A impossibilidade de testar o código desenvolvido resultará em penalizações na avaliação. Todo o código deverá ser devidamente comentado.

Avaliação

Cada grupo deve entregar um ficheiro README.pdf e o código-fonte desenvolvido, bem como realizar uma demonstração da aplicação. A submissão deverá ser em formato ZIP, e o nome do ficheiro deverá ser:

`PFL_TP1_`**`#GRUPO`**`.zip`

em que **`#GRUPO`** é a designação do grupo. Exemplo: `PFL_TP1_G1_12.zip`

O ficheiro ZIP deverá conter um ficheiro **README.pdf** e os ficheiros de **código-fonte** Haskell, o quais deverão ser **devidamente comentados**. Os ficheiros do código-fonte, Fib.hs e BigNumber.hs (e outros eventuais ficheiros auxiliares), deverão estar ao mesmo nível que o README.pdf (e não num subdiretório). O ficheiro ZIP não deverá conter nenhum diretório, todos os ficheiros devem estar na raíz.

O ficheiro README.pdf deverá conter:

- a descrição de vários casos de teste para todas as funções;

- uma explicação sucinta do funcionamento de cada função;
- as estratégias utilizadas na implementação das funções da alínea 2;
- a resposta à alínea 4.

As demonstrações serão combinadas com o docente de cada turma prática.