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

Prova I - Programação Funcional - 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. Responda todas as questões referentes ao problema 1, no arquivo do problema 1. Faça o mesmo para as questões referentes ao problema 2, e assim por diante.

- **Para cada problema, inclua chamadas para suas funções, assim demonstrando o seu funcionamento.**
- Lembre-se: você pode criar funções auxiliares para ajudar na solução das questões.
- A legibilidade, organização do código, bem como o uso de comentários também serão considerados na avaliação.
- Note que a somatória dos pontos passa de 10, porém mesmo acertando todos os problemas, a nota máxima será 10.

1. **(1.0)** Crie uma função que receba um valor n como parâmetro e retorne a soma dos n primeiros números naturais **ímpares** (de 1 até n (inclusive)).
2. **(1.0)** Crie uma função que receba um valor n como parâmetro e retorne o n-ésimo número de Fritz. O n-ésimo número de Fritz pode ser descoberto utilizando a seguinte fórmula:

$$F_n = \begin{cases} 0 & se\ n = 0 \\ 1 & se\ n = 1 \\ F_{n-1} + 3F_{n-2} & caso\ contrario \end{cases}$$

   Os números de Fritz iniciam com os seguintes números: 0, 1, 1, 4, 7, 19, 40, 97, 217, 508, 1159, 2683, 6160, 14209, ...

3. Motifique o arquivo `arvore.hs` (disponível no Moodle) de forma a adicionar novas operações à nossa árvore:

   A (1.5): Crie uma função com a seguinte assinatura: `profundidadeElemento :: Arvore -> Int -> Int`, a qual recebe um número e deve retornar a profundidade em que o elemento foi encontrado na árvore. Considere que o nó raiz está na profundidade 0 (zero). Se houverem mais ocorrências do mesmo número, retorne qualquer uma das profundidades em que ele foi encontrado.

   B (1.5): Crie uma função com a seguinte assinatura: `folhasImpares :: Arvore -> [Int]`, a qual recebe uma árvore como parâmetro e deve retornar uma lista dos valores das folhas da árvore que são números **ímpares**. Uma folha da árvore é um nó que não possui filhos, ou seja, a árvore da esquerda e da direita é nula.

   C (2.0): Crie uma função com a seguinte assinatura: `elementosRepetidos :: Arvore -> Int`, a qual deve retornar a quantidade de números repetidos na árvore. Não utilize funções prontas do Haskell para esta questão. Se um número está repetido várias vezes, conte-o uma única vez.

4. Para as questões abaixo, você deverá criar os tipos abaixo:

   - Crie um tipo (**data** ou **type**) para um ponto 2D
   - Crie um tipo (**data** ou **type**) para a forma geométrica Circulo, o qual deve ser descrito por um ponto 2D, que indicará o centro do círculo, e um Float para indicar o raio.

   A (2.0): Crie uma função com a seguinte assinatura: `dentro :: Circulo -> [Ponto] -> [Ponto]`, a qual recebe um círculo e uma lista de pontos e retorna a lista de pontos que estão dentro do círculo (considere que estando na linha, significa que está dentro). Caso não houver nenhum ponto dentro do círculo, retorne uma lista vazia.

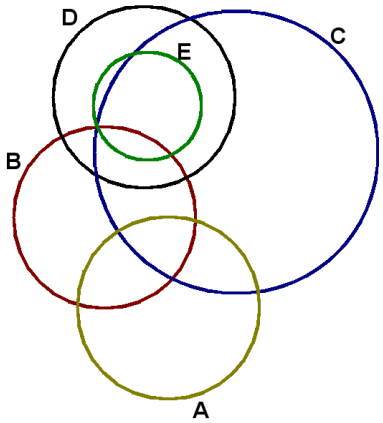

Figura 1: Intersecção de círculos.

B (3.0): Crie uma função com a seguinte assinatura: `interseccao :: [Circulo] -> Int`, a qual recebe uma lista de círculos e retorna a quantidade máxima de círculos que se intersectam **todos** uns com os outros (note que dois círculos que apenas se tocam não estão intersectando). No caso da Figura 1, a quantidade máxima de círculos onde todos intersectam entre si é 4. **Explique seu raciocínio em um comentário para este problema.**