UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

PROFESSORES: GIOVANNY F. L. PALMA e LEILA M. A. SILVA

PRIMEIRA PROVA DE PROGRAMAÇÃO FUNCIONAL

INSTRUÇÕES: Esta prova tem **2h de duração e 30 min de tolerância** para o envio pelo **Google Classroom**. **Cada questão vale 2,0 pontos.** Gere um único arquivo contendo as respostas textuais de todas as questões. Insira seu **nome completo** e **matrícula** no cabeçalho da sua resposta. O arquivo com as soluções deve ser em formato **PDF**. O nome de seu arquivo deve possuir o formato **SeuNomeUltimoSobrenome-P1.pdf**. Por exemplo, no caso da professora da disciplina seria LeilaSilva-P1.pdf. As questões podem ser feitas no editor de texto de sua preferência.

IMPORTANTE: Nesta prova você só pode utilizar funções pré-definidas do Prelude e da biblioteca Data.Char e compreensões. Caso você seja um aluno que tenha um conhecimento de Haskell anterior ao curso, não poderá usar recursão e/ou funções de alta ordem na solução das questões, nem funções pré-definidas de outras bibliotecas de Haskell, pois o objetivo desta prova é verificar o conhecimento adquirido com o conteúdo ministrado até o momento da avaliação.

1. Escreva uma função que verifica se a soma dos dois dois últimos dígitos de um inteiro positivo é par. Se o número tiver somente um único dígito, complete com zero à esquerda.

2. Considere uma lista de tuplas em que o primeiro elemento da tupla é o nome de uma pessoa, o segundo é o gênero, o terceiro o ano de nascimento e o quarto o estado civil. O gênero admite os valores ‘F’, ‘M’ e ‘X’, os quais denotam, respectivamente, os gêneros feminino, masculino e demais gêneros. O estado civil admite os valores ‘C’, ‘S’, ‘V’ e ‘O’, denotando, respectivamente, os estados civis casado, solteiro, viúvo e outros estados civis. Declare tipos para todos os dados e elabore uma função que recebe como entrada esta lista de tuplas, o ano corrente e uma faixa de idades e gera como resultado uma lista de nomes de pessoas que pertencem à faixa de idade informada. A faixa de idade é um par cartesiano com duas idades, que são os valores inferior e superior da faixa desejada.

3. Elabore uma função para calcular a seguinte soma, em que os valores de *a* e *b* são fornecidos como parâmetros da função:

$$\sum_{x=a}^{b} \frac{1}{x} = \frac{1}{a} + \frac{1}{(a+1)} + \frac{1}{(a+2)} + \ldots + \frac{1}{b}$$

4. Elabore uma função `zerosPrecedem` tal que dada uma lista de 0s e 1s, contendo pelo menos um elemento 0 e um elemento 1, devolve `True` se todos os elementos zeros precedem todos os elementos 1s na lista. Caso contrário, devolve `False`. Além disso, se a lista fornecida pelo usuário não atender ao exigido no enunciado, você deve informar que a lista fornecida é inválida, usando error. Por exemplo,

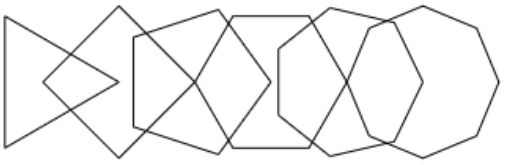
   `zerosPrecedem [1,1,0,1,0]` devolverá `False`

   `zerosPrecedem [0,0,0,1]` devolverá `True`

   `zerosPrecedem [0,0,0,0]` a mensagem "`lista invalida`" será exibida

5. Defina uma função que crie uma `Picture` contendo uma fila horizontal de polígonos regulares como esta aqui

   A função terá três entradas: o número total de polígonos regulares na `Picture`, o raio (comum a todos os polígonos) e a distância entre os centros de cada par consecutivo de polígonos. Observe que, na figura anterior, esta distância é um pouco maior que o dobro do raio. Se a distância escolhida for igual ao raio de cada polígono, a função deverá construir esta figura



   Para a construção de um único polígono regular você pode usar a seguinte definição vista em sala de aula.

```
poligonoRegular :: Int -> Double -> Picture
poligonoRegular n r =
     polygon [ (r * cos (i * theta), r * sin (i*theta))
               | i <- [0 .. fromIntegral n - 1] ]
  where
    theta = 2*pi / fromIntegral n
```