UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

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

TERCEIRA PROVA DE PROGRAMAÇÃO FUNCIONAL

INSTRUÇÕES: Esta prova tem **2:30h de duração, incluindo o tempo de envio** pelo **Google Classroom**. 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-P2.pdf**. Por exemplo, para o nome da professora Leila seria LeilaSilva-P2.pdf. As questões podem ser feitas no editor de texto de sua preferência. **Para cada 5 minutos de atraso que exceder o tempo de prova estipulado o aluno será descontado de -2,0 pontos.**

IMPORTANTE: Nesta prova você **deve usar funções de alta ordem** e funções pré-definidas em qualquer biblioteca Haskell. Você **não poderá usar compreensões, nem recursão apenas, quando funções de alta ordem puderem ser utilizadas para realizar a mesma tarefa**, pois o objetivo desta prova é verificar o conhecimento adquirido com o conteúdo ministrado na terceira unidade do curso.

Nome do Aluno:
Matrícula:

Escolha apenas 4 questões para resolver. Cada questão vale 2,5 pontos.

1. Suponha uma lista `xs` de Booleanos. Defina a função `meuAnd`, tal que dado `xs`, devolve uma lista em que faz a conjunção (&&) do primeiro e do segundo elemento, do terceiro e do quarto e de forma geral dos elementos das posições *i* e *i+1, i* ímpar, enquanto for possível. **Nesta questão é proibido o uso de compreensão e recursão.** Por exemplo,

   `meuAnd [True, False, True, True, False]` devolverá `[False, True]`

   `meuAnd [True]` devolverá `[]`

2. Defina um tipo algébrico polimórfico para árvore binária e usando este tipo elabore uma função para, dada uma árvore, verificar se uma árvore é de busca. Lembrando que uma árvore binária é de busca se e somente se para todo nó não vazio com valor *v* todos os valores da subárvore esquerda são menores ou iguais a *v* e todos os valores da subárvore direita são maiores que *v*.

3. Considere a seguinte definição.

```
f p xs = foldr g True xs
  where
    g x y = p x && y
```

Qual é o tipo mais geral de `f`?

Qual é o tipo mais geral de `g`?

O que calcula a função `f`?

Redefina `f` sem usar definição local. Para isto use expressões lambda. A redefinição de `f` deve ser *point-free*, sem explicitar o último argumento da função.

4. Elabore um programa que faz:

   a. Lê um número *n;*

   b. Lê *n* números inteiros e forma uma lista *A;*

   c. Lê um número *m;*

   d. Lê *m* números inteiros e forma uma lista *B;*

   e. Calcula a lista de interseção de *A* e *B;*

   f. Imprime a lista de interseção no formato abaixo.

      Por exemplo,

      Entrada:

      3

      10

      8

      30

      4

      10

      30

      15

      40

      Saída:

      [10, 30]

5. Prove que:

```
sum (xs ++ dobrolista ys) = sum xs + 2 * sum ys
```

considerando as seguintes definições de funções:

```
sum []  = 0                                   (s.1)

sum (y:ys) = y + sum ys                       (s.2)

[] ++ zs = zs                                 (++.1)

(w:ws) ++ zs = w:(ws++zs)                     (++.2)

dobrolista [] = []                            (dL.1)

dobrolista (w:ws) = 2*w: dobrolista ws        (dL.2)
```