UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

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

SEGUNDA PROVA DE PROGRAMAÇÃO FUNCIONAL

INSTRUÇÕES: Esta prova tem **2:30h de duração, incluindo o tempo de 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-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 **só pode utilizar recursão** e funções pré-definidas do Prelude e da biblioteca Data.Char **que foram dadas nas aulas disponibilizadas nos slides até a semana 9**. Você **não poderá usar compreensões, outras funções das bibliotecas do Haskell que não estejam nos slides até a semana 9, e/ou funções de alta ordem** na solução das questões, pois o objetivo desta prova é verificar o conhecimento adquirido com o conteúdo ministrado até a segunda unidade do curso.

Nome do Aluno:
Matrícula:

1. Elabore uma função recursiva que receba uma lista de inteiros e identifique se algum elemento da lista é negativo. Por exemplo:
   para a entrada `[2,3,4]` a função deve retornar `False`;
   para a entrada `[2,-1,0]` a função deverá retornar `True`.
2. Dadas duas listas de inteiros, representando dois conjuntos *A* e *B*, elabore uma função que retorne a lista que representa a diferença destes conjuntos, ou seja o conjunto formado pelos elementos de *A* que não estão em *B*. Por representarem conjuntos, estas listas não possuem elementos repetidos.
3. Defina uma função que receba duas listas de inteiros ordenadas ascendentemente, ambas sem elementos repetidos, e calcule a união das duas listas. O resultado não deve conter elementos repetidos e deve estar ordenado. Sua solução deve se basear em um dos algoritmos de ordenação vistos em aula. Diga qual escolheu.
4. Usando a técnica de **pedir mais informação ao amigo**, defina uma função que retorne as posições em que se encontra o maior elemento de uma lista. Note que o maior pode ocorrer em várias posições. **Sua solução não pode utilizar nenhuma função pré-definida no Prelude nem definir função auxiliar que faça o mesmo que alguma destas.**

5. Defina uma função que, dada uma lista de números Double, todos positivos, e uma lista de cores, ambas listas do mesmo tamanho, construa um diagrama de barras. Haverá uma barra por cada elemento na lista de números, sendo que o tamanho de cada barra será proporcional ao valor na primeira lista e terá a cor correspondente à dada na segunda lista. Por exemplo, para as listas `[12, 34, 5, 8, 40]` e `[red, blue, green, gray, yellow]`, a Picture é