TRABALHO 3 – RECURSIVIDADE E LISTAS

Esta avaliação é individual.

Este trabalho corresponde a alínea Exercícios e Trabalhos do plano de ensino e como tal terá peso na avaliação da RA1.

Para que seu trabalho seja avaliado você deverá postar, no ambiente virtual de aprendizagem, na área reservada para este fim, dois links com a solução do seu trabalho. O primeiro destes links deve apontar para arquivos no ambiente repl.it[1], onde será possível executar sua solução. O segundo link deve apontar para arquivos contendo os mesmos códigos, entretanto este código deve estar hospedado no ambiente Github. **Links que apontem para códigos diferentes em ambientes diferentes serão provocarão o zeramento do trabalho.**

Todos os códigos enviados devem conter o enunciado que está sendo resolvido, na forma de comentário, em pelo menos um dos arquivos enviados. **Todos os arquivos de código devem conter, na primeira linha, em forma de comentário, o nome completo do aluno.**

Você deve ler todo este documento antes de começar e considerar o seguinte código de ética:

I. Você poderá discutir todas as questões com seus colegas de classe, professores e amigos. Poderá também consultar os livros de referência da disciplina, livros na biblioteca, virtual ou não, e a internet de forma geral e abrangente nos idiomas que desejar. Contudo o trabalho é seu e deverá ser escrito por você. Cópia é plágio.

Para conseguir os pontos referentes a esta avaliação você deverá seguir as instruções apresentadas no item Enunciado deste documento.

OBJETIVO

Pesquisar e praticar. Pesquisar os conteúdos que irão complementar o material apresentado em sala ou nos livros sugeridos na ementa e praticar estes mesmos conceitos. Esta é uma oportunidade para aprimorar sua formação e se destacar profissionalmente.

METODOLOGIA UTILIZADA

Você pode e deve fazer uso das ferramentas disponíveis online para busca de informações. Pode e deve consultar seus colegas de classe e **pode mandar dúvidas para o professor até o dia anterior a data limite de entrega**. Observe as seguintes indicações:

- ao usar o Google, lembre-se de usar os comandos de busca para melhorar a qualidade dos resultados.
- faça sua pesquisa considerando apenas arquivos no formato pdf, disponíveis em instituições de ensino, em qualquer idioma, publicados nos últimos 5 anos;
- Caso seja necessário regidir algum texto para este Trabalho, o texto deverá ser escrito segundo as normas da ABNT. Use apenas as normas que fazem sentido para o trabalho de pesquisa que você está desenvolvendo. Preocupe-se com as fontes, espaçamentos, formato de parágrafos e citações.

[1] Alternativamente você pode usar: Online Haskell Compiler (tutorialspoint.com)

Por fim, lembre-se que nenhum trabalho, exercício, ou pesquisa científica, ou acadêmica, admite qualquer tipo de plágio e que todos os conceitos que você trouxer para o seu trabalho deverão ser acompanhados da citação correta. Lembre-se também que todos os trabalhos enviados passarão por um sistema de avaliação de plágio e que **trabalhos contendo plágio serão zerados**.

ENUNCIADO

A seguir estão listados 10 problemas que você deverá resolver utilizando a linguagem Haskell.

A resolução de todos estes problemas deverá constar em uma única folha de código Haskell, disponível online segundo as regras explicitadas anteriormente neste documento.

Todos os problemas devem ser resolvidos com a criação de funções usando a linguagem Haskell. E **todas estas funções precisam ter seus tipos explicitados**. Você não pode usar a inferência de tipos.

Antes de cada solução, deverá existir um comentário contendo o enunciado da questão que aquela função específica resolve. Para isso você DEVE COPIAR E COLAR o enunciado de cada questão na forma de comentário no código Haskell.

Como todas as questões serão resolvidas por funções. A prova de que a função resolve uma determinada questão deve ser feita por meio da chamada desta função no *main* do módulo que você criar para a solução dos problemas. **Lembre-se você precisa testar a função com casos positivos e negativos, sempre que for necessário.**

Os testes de cada função devem ser realizados pela chamada da função e a impressão, no terminal do resultado obtido. Todas as impressões devem seguir o seguinte padrão:

Func. 1: entrada:2; resultado:3

Cada teste de função deve imprimir uma, e somente uma linha no terminal.

Se, por algum motivo, os dois serviços online usados para a excução do trabalho estejam indisponíveis você deverá anexar à sua resposta, um link para um documento online (pdf) contendo uma prova da indisponibilidade destes serviços. A prova pode ser a captura das telas de erro onde seja possível identificar data e hora da indisponibilidade. Neste caso, deve ser enviado apenas o link do Github.

Cada um dos problemas a seguir tem valor 0,15 e todas as funções desenvolvidas que não fizerem o que foi solicitado no enunciado ou, apresentarem qualquer erro, serão anuladas.

1. Escreva uma função para o cálculo dos números da sequência de Fibonacci, utilizando Haskell.
2. Um dos primeiros algoritmos documentados é o algoritmo para o cálculo do Maior Divisor Comum (MDC) de Euclides publicado por volta do ano 300 AC. Podemos simplificar este algoritmo dizendo que dados dois inteiros A e B, o MDC entre eles será dado pelo valor absoluto de A se B=0 e pelo MDC entre B e o resto da divisão de A por B se B>0. Escreva uma função para o cálculo do MDC entre dois números inteiros positivos, usando o algoritmo de Euclides conforme apresentado aqui, utilizando Haskell.
3. Escreva uma função recursiva que dado um número inteiro n, devolva a soma dos dígitos deste número. Exemplo: dado 1234 a função deverá devolver 10. Utilizando Haskell e recursividade.

4. Escreva uma função que devolva a soma de todos os números menores que 10000 que sejam múltiplos de 3 ou 5.
5. Escreva uma função que, recebendo uma lista de inteiros, apresente a diferença entre a soma dos quadrados e o quadrado da soma destes inteiros, usando recursividade.
6. O Crivo de Eratóstenes não é o melhor algoritmo para encontrar números primos. Crie uma função que implemente o Crivo de Euler (Euler's Sieve) para encontrar todos os números primos menores que um determinado inteiro dado.
7. Nem só de Fibonacci vivem os exemplos de recursão. Escreva uma função que devolva todos os números de uma sequência de Lucas (2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123) menores que um inteiro dado.
8. Escreva uma função, chamada *aoContrario* em Haskel para reverter uma lista. Dado [1,2,3] devolva [3,2,1].
9. Escreva uma função chamada *somaRecursiva* que recebe dois valores inteiros e devolve o produto destes valores sem usar o operador de multiplicação.
10. Escreva uma função chamada comprimento que receba uma lista de inteiros e devolva o comprimento desta lista. Observe que você não pode usar nenhuma função que já calcule o comprimento de uma lista.

RUBRICAS DE AVALIAÇÃO

As notas serão atribuidas segundo as seguintes regras:

I. Todas as funções estão corretamente definidas, testadas e as saídas estão corretamente formatadas: $\mathbf{Nota = 10;}$

II. Todas as funções estão corretamente definidas e testadas mas, as saídas não estão corretamente formatadas: $\mathbf{Nota = 10 - (1 * Nº\ Erros);}$

III. Algumas funções não foram implementadas, ou não foram testadas, ou ainda algumas das impressões não estão corretamente formatadas: $\mathbf{Nota = 10 - (1 * Nº\ funções\ faltantes + 1 * Nº\ Erros\ de\ Formatação)}$;

IV. Não foi possível rodar todas as funções, ou o código enviado não atende o enunciado: $\mathbf{Nota = 0\ (zero)}$;

V. Programa não pode ser acessado pelo professor: $\mathbf{Nota = 0\ (zero)}$.

CUIDADOS QUE VOCÊ PRECISA TOMAR:

As regras de perda de ponto por entrega fora do prazo, constantes no plano de ensino, se aplicam a este trabalho.

Programas com códigos idênticos serão zerados.

Certifique-se que o seu código pode ser acessado por alguém além de você. Cada ambiente tem uma regra diferente de postar o código de forma que ele seja acessível. Para não ter problemas, depois que terminar e antes de postar os códigos. Acesse os links usando outra indentidade no seu navegador *web* ou peça para algum colega acessar seus links. **Programas com código inacessível serão zerados.**