Semana 1 - Lista de exercícios

(Esta lista é para ser feita individualmente)

Cada exercício (ou letra) possui a informação de qual é o seu valor em pontos, seguindo modelo a seguir: (valor: x pontos).

A nota da lista será data por:

L = min(10, P/5)

Onde:

L = Nota da lista

P = Pontos obtidos na lista

Lembrando que está lista vale 50% da nota final do curso.

Envie as resposta das questões de 1 a 8 em um arquivo pdf junto aos dois arquivos de código da questão 9, compactados em um arquivo zip. O nome do arquivo zip deve ser seu nome completo.

1) Construa o *list comprehension* que gere:

a) [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]

(valor: 1 ponto)

b) uma lista de 'a' a 'z' sem as vogais.

(valor: 1 ponto)

c) uma lista de 0 a 50 sem os números 2, 7, 13, 35 e 42.

(valor: 1 ponto)

d) [500.0, 250.0, 125.0, 62.5, 31.25, 15.625, 7.8125, 3.90625, 1.953125, 0.9765625]

(valor: 1 ponto)

e) uma lista com todas as coordenadas dde um tabuleiro de damas 8x8 ([('a',1), ('a',2), ('a',3) ... ('h',7), ('h',8)])

(valor: 1 ponto)

2) Crie as funções da forma como solicitado:

a) Crie um função recursiva que verifique se um elemento esta presente em uma lista, deixe explicita a assinatura da função.

(valor: 1 ponto)

b) Crie um função que receba 3 valores numéricos, de qualquer tipo, e considerando que cada valor é a medida de um lado de um triângulo, retorne um valor do tipo Bool informando se estas são medidas validas para formar um triângulo.

(valor: 1 ponto)

c) Crie uma função recursiva que tenha dois parâmetros númericos, de qualquer tipo, e calcule a potenciação, o primeiro parâmetro sendo a base e o segundo o expoente.

(valor: 1 ponto)

3) Siga as intruções a baixo na sequência:

a) Crie um tipo Cor que possua um *value constructor* de mesmo nome que carrege 3 valores do tipo Int.

(valor: 1 ponto)

b) Cada um dos valores do tipo Int são relativos aos valores RGB (red, green, blue), pra cada um desses valores crie (manualmente) uma função de projeção, o nome das funções devem ser red, green e blue, respectvamente para cada um dos valores do tipo Int.

(valor: 1 ponto)

c) Ao invés de criar as funções de projeção manualmente, crie o tipo Cor utilizando record syntax.

(valor: 1 ponto)

d) Crie uma função com o nome somaCor, que combine dois valores do tipo Cor, de forma que o resultado deve ser um novo valor, também do tipo Cor, onde os valores relativos ao RGB (red, green e blue), são relativos a soma dos valores RGB dos parâmetros da função, respectivamente. Caso algum valor relativo relativo ao valores RGB do resultado seja maior que 255, deve-se colocar 255 no lugar.

(valor: 2 pontos)

e) Crie um operador <+> que faça a mesma coisa que a função somaCor, utilizi-se do conceito de currying para definir o operador. (Os símbolos de maior e menor fazem parte do operador)

(valor: 1 ponto)

f) Crie uma instância de Monoid para o tipo Cor, onde a operação bínaria seja o operador <+> e o elemento nêutro seja um valor do tipo Cor que contem os valores RGB como 0.

(valor: 2 pontos)

e) A instância de Monoid criada é valida considerando as leis dos Monoids? Explique. (valor: 2 pontos)

4) Siga as intruções abaixo em sequência:

a) Crie um tipo Cofre que possua uma *type variable*, e um *value constructor* de mesmo nome que o tipo, que carregue um valor que seja um lista do tipo da *type variable*.

(valor: 2 pontos)

b) Crie uma instância do *typeclass* Functor para esse tipo.

(valor: 2 pontos)

c) Crie uma instância do *typeclass* Applicative para esse tipo.

(valor: 2 pontos)

5) Siga as intruções a baixo na sequência:

a) Crie o tipo Automovel que possua dois *value constructors*, um para representar o carro, e o outro para representar a moto. (os *value constructors* não carregam nenhum valor)

(valor: 1 ponto)

b) Crie um tipo Veiculo que possua um *value constructor* de mesmo nome que carregue 2 valores, o primeiro de nome automovel do tipo Automovel, o segundo de nome placa do tipo String. (Utilize record syntax)

(valor: 1 ponto)

c) Crie um typeclass EhCarro, para tipos de kind *, que possua uma função com nome ehCarro que receba um valor do tipo do type parameter do typeclass, e retorne um valor do tipo Bool informando se é um carro.

(valor: 2 pontos)

d) Crie uma instância de EhCarro para o tipo Veiculo.

(valor: 1 ponto)

e) Crie uma instância de EhCarro para o tipo Int (Nenhum número é um carro).

(valor: 2 ponto)

6) Seguindo os aximos de Peano (cada letra vale 1 ponto):

a) Zero é um número natural, e todo sucessor de um número natural também é um número natural. Crie o tipo Natural, que possua dois *value contructors*, um para representar o valor zero, e o outro para representar a sucessão de um valor do tipo Natural. (Dica: este é um tipo recursivo)

(valor: 2 pontos)

b) Com o conceito de sucessão contido na definição do tipo Natual, podemos representar o valor 1 como o sucessor de zero, o valor 2 como o sucessor do sucessor de zero, e assim por diante. Crie uma função somaNatural, que receba dois parâmetros do tipo Natural que você criou e retorne um valor do tipo Natural que seja a soma dos dois parâmetros. (valor: 2 pontos)

c) Crie uma função que converta de Natural para Int.

(valor: 2 pontos)

7) Crie uma função que receba 3 parâmetros, sendo eles duas funções (referenciadas aqui no enunciado como f e g) e um valor (referenciado aqui no enunciado como x), e retorne uma tupla de duas posições contendo na primeira posição o resultado de f composta em g aplicado em x e na segunda o resultado de g composta em f aplicado em x. Construa a função da forma mais genérica que for possível de criar uma função com essas caracteristicas e deixe explicita a assinatura da função. Faça uma breve explicação sobre a forma como vc pensou para criar essa função.

(valor: 3 pontos)

8) Crie um tipo JoKenPo que possua três *value constructors*, representando as jogadas predra, papel e tesoura. Crie um tipo Resultado que possua três *value constructors*, representado os resultados da vitória do jogador 1, a vitória do jogador 2 e o empate. Crie uma função jogar, que recebe duas jogadas, do tipo JoKenPo, e retorne o resultado, do tipo Resultado. Considere o primeiro parâmetro como jogador 1 e o segundo como jogador 2.

(valor: 3 pontos)

9) Crie os programas solicitados abaixo (crie qualquer tipo, função ou qualquer outras coias que julgar necessario para a implementação do seu programa). Envie cada uma das letras como um arquivo haskell (.hs), para que possa ser compilado e executado:

a) Para construir um foguete, e ir até a lua, são necessários vários componentes, quanto mais componentes, mais combustivel é necessário para realizar a viagem. Para calcular quanto combustível é necessario para cada componente você divide o peso do componente por 3 e aredonda o valor para baixo (número inteiro) e depois subtrai 2, por exemplo, para um componente que pesa 13 quilos, será necessário 2 kilos de combustível. Faça um programa que leia um arquivo de nome "componentes.txt" no mesmo dirétorio de execução do programa e calcule o total de combustível necessario para realizar a viagem. O arquivo "componentes.txt" possui varias linhas, cada linha possui apenas um número inteiro relativo ao peso de um componente que está no foguete.

(valor: 5 pontos)

b) O programa anterior tem um problema, ele considera apenas o peso dos componentes, porém, é necessário considerar quanto de combustível é necessario para lear o combusti´vel de cada componente. Para isso é necessário calcular quantos quilos de combustível é necessário para um componente, e depois qualquer quanto de combustivel é necessário para o combustivel que foi calculado, e fazer sucessivamente até que não seja mais necessário adicionar mais combustivel. Faça um programa igual ao anterior, mas considere essa informação que foi passada para alterar o calculo de combustível.

(valor: 7 pontos)