UNIVERSIDADE CESUMAR - UNICESUMAR

Curso: Engenharia de Software	**Série:** 6S	**Turma:** A	**Turno:** Noite
Professor(a): Thiago Bussola da Silva	**Horário:**		
Acadêmico (a): Lucas Lorenzão de Pieri Poi			**RA: 21010402-2**
Disciplina: Paradigmas de Programação			**Data:26/09/23**

Prova	Prova Prática	Atividades de estudo programadas (AEP)	Prova integrada	Nota final do bimestre

INSTRUÇÕES PARA REALIZAÇÃO DA PROVA:

⇒ Os dados do cabeçalho deverão ser preenchidos com letra maiúscula. E as questões deverão ser respondidas com letra legível.
⇒ É vedado, durante a prova, o porte e/ou o uso de aparelhos sonoros, fonográficos, de comunicação ou de registro eletrônico ou não, tais como: notebooks, celulares, tablets e similares.
⇒ A prova é individual e sem consulta, deverá ser respondida a caneta azul ou preta. Prova escrita a lápis não dá direito à revisão. Não é permitido o uso de corretivo.
⇒ É obrigatória a permanência do acadêmico 1 (uma) hora em sala de aula após o início da prova.
⇒ Não será permitida a entrada na sala de aula após 10 minutos do início da prova.
⇒ É obrigatória a assinatura da lista de presença impressa na qual constam RA, nome e curso.
⇒ O valor de cada questão está ao lado da mesma.
⇒ Todas as respostas devem constar no espaço destinado e autorizado pelo professor, à resposta.
⇒ Em caso de qualquer irregularidade comunicar ao Professor ou fiscal de sala.
⇒ Ao término da prova, levante o braço e aguarde o atendimento do professor ou do fiscal.

1ºbim.		2ºbim.		1ªsub.		2ªsub.		1ºsem.		2º sem.	

QUADRO PARA O PROFESSOR - REGISTRO DE NOTAS	
Questão 1	
Questões 2	
Questão 3	
Questão 4	
Questão 5	
Questão 6	
Questões 7	
Questões 8	
Questão 9	
Questão 10	

Instruções - Leia com atenção!

Preencha os campos do cabeçalho da prova

Regras para a prova.

Os únicos sites que você pode acessar para consultar suas dúvidas sobre sintaxe são:

https://elixirschool.com/pt/lessons/basics/documentation

https://elixir-lang.org/docs.html

O uso de qualquer outro site, chat GPT, Github está **proibido**, caso o aluno acesse outra fonte de pesquisa a prova será zerada.

Compiladores: Será permitido o uso de compiladores online para que você possa validar a implementação das soluções propostas para os exercícios. Você está autorizado a utilizar os seguintes compiladores:

https://www.tutorialspoint.com/execute_elixir_online.php

https://onecompiler.com/elixir

O uso do **Replit não está autorizado** e caso o aluno acesse essa ferramenta a prova será zerada.

Você pode criar arquivos .exs para a resolução da prova e fazer o zip para enviar eles. Ou você pode copiar o código de resposta e colar abaixo da pergunta correspondente no arquivo .docx

Você pode converter sua prova para pdf ao enviar, lembre-se de enviar os arquivos .exs ou de colocar as respostas na prova para a entrega.

Caso você não entregue o arquivo .docx / pdf e os .exs (caso tenha seja de sua preferência) a prova será zerada.

Questão 1 - [1 ponto] - Explique a diferença entre funções puras e funções de ordem superior em programação funcional. Dê exemplos de cada uma.

As funções puras são funções mais "simples" que com os mesmos valores de entrada sempre receberão os mesmos valores de saída, isto é se temos uma função: soma(a, b) que em sua estrutura soma a + b, sempre dando os mesmos valores teremos os mesmos resultados.
Já as funções de ordem superior aceitam funções como variáveis, isto é, podendo aceitar funções como parâmetro, podendo retornar funções e entre outros. Um exemplo disso seria uma função que soma e adiciona 1: somaAddUm(soma, a, b) que em sua estrutura tem a chamada da função anônima soma acrescentando 1 -> func.(a, b) + 1.

Questão 2 - [1 ponto] - Discorra sobre as vantagens da linguagem de programação elixir e em que tipo de projeto ou cenário devemos optar pelo uso dessa tecnologia.

Devido ao Elixir ser uma linguagem de programação funcional que quebra os problemas em problemas menores, sem efeitos colaterais e com menos bugs. Além dela ser imutável, ter uma facilidade com ambientes que escalam e oferece concorrência, além disso é focada em funções simples (puras), além de serem "first class citizens" o que permite a reutilização delas ao decorrer da aplicação.

Questão 3 - [0,5 pontos] - Escreva uma função que verifique se um número é par.

```
import Integer

data = IO.getn("Digite um numero para saber se é par ou impar:")
number = String.to_integer(data)
IO.puts(is_even(number))
```

Questão 4 - [0,5 pontos] - Implemente uma função que calcule o dobro de cada elemento em uma lista.

```
lista = [1, 2, 3, 20]
lista_dobrada = Enum.map(lista, fn x -> x * 2 end)
IO.inspect(lista_dobrada)
```

Questão 5 - [0,5 ponto] - Crie uma função que retorne o último elemento de uma lista.

```
lista = [1, 2, 3, 20, 12]
ultimo_lista = List.last(lista)
IO.inspect(ultimo_lista)
```

Questão 6 - [1 ponto] - Implemente uma função que calcule o fatorial de um número usando recursão.

```
defmodule Math do
   def fatorial(1), do: 1
   def fatorial(a) do
      a * fatorial(a-1)
   end
end

IO.inspect Math.fatorial(5)
```

Questão 7 - [1 ponto] - Escreva uma função que aplique uma função passada como argumento a cada elemento de uma lista.

```
defmodule Math do
   def executeFuncToEach(func, list) do
      Enum.map(list, func)
   end
end

lista = [1, 2, 3, 20, 12, 15]
addOne = fn x -> x + 1 end

IO.inspect(Math.executeFuncToEach(addOne, lista))
```

Questão 8 - [1 ponto] - Escreva uma função que filtre os elementos de uma lista com base em uma função de filtro passada como argumento.

```
defmodule Math do
   def executeFilterToEach(func, list) do
      Enum.filter(list, func)
   end
end

lista = [1, 2, 3, 20, 12, 13, 7]
filterOdd = fn x -> rem(x, 2) != 0 end
```

```
IO.inspect(Math.executeFilterToEach(filterOdd, lista))
```

Questão 9 - [1 ponto] - Crie uma função que gere os primeiros "n" números da sequência de Fibonacci.

```
defmodule Math do
   def fibonacci(0), do: 0
   def fibonacci(1), do: 1
   def fibonacci(a) do
      fibonacci(a-1) + fibonacci(a-2)
   end
end

IO.inspect(for x <- 1..10, do: Math.fibonacci(x)
```

Questão 10 - [2,5 ponto] - Crie uma função que calcule a média de uma turma.
O exemplo abaixo demonstra como são passadas as notas dos alunos pertencentes a uma turma.
Você deve utilizar o método reduce para calcular a média.

```
notas_da_turma = [
  {"Alice", [9.5, 8.0, 7.5]},
  {"João", [8.0, 7.0, 6.5]},
  {"Pedro", [9, 9.5, 9.0]},
   {"Lucas", []},
]
```

Dicas: Utilize map ou flatmap para extrair todas as notas dos alunos em uma única lista

```
defmodule CalcularMedias do
   def calculo(lista) do
      lista_notas = Enum.map(lista, fn {_alunos, notas} -> notas end)

      notas_validas = Enum.filter(lista_notas, fn x -> length(x) > 0 end)

      lista_limpa = List.flatten(notas_validas)

      media_turma = Enum.reduce(lista_limpa, fn (x, acc) -> x + acc end)

      quantidade_notas = length(lista_limpa)

      media_turma / quantidade_notas
   end
end
```

```
notas_da_turma = [
  {"Alice", [9.5, 8.0, 7.5]},
  {"João", [8.0, 7.0, 6.5]},
  {"Pedro", [9, 9.5, 9.0]},
   {"Lucas", []},
]

IO.inspect(CalcularMedias.calculo(notas_da_turma))
```