UNIVERSIDADE CESUMAR - UNICESUMAR

Curso: Engenharia de Software	**Série:** 6S	**Turma:** A	**Turno:** Noite
Professor(a): Thiago Bussola da Silva	**Horário:**		
Acadêmico (a): Allan Ogawa			**RA: 21105670-2**
Disciplina: Paradigmas de Programação			**Data: 26/09/2023**

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.

funções de ordem superior são funções que podem ter como parâmetro uma função ou que podem retornar outras funções, enquanto a função pura é a função simples, que tem como parâmetros e retornos variáveis ou constantes. Um exemplo de função de ordem superior é uma função recursiva, como a função de fibonacci, enquanto um exemplo de função pura pode ser a função de soma, onde são passado 2 números como parâmetro e retorna a soma deles.

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.

Elixir é usado pela sua imutabilidade, funções puras e escalabilidade de processos, onde rodam centenas de milhares de processos simultaneamente, fornecendo uma melhor funcionalidade para a máquina. Ele é bastante usado em projetos de desenvolvimento web, software incorporado, aprendizado de máquina, pipeline de dados e processamento de multimídias

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

```
defmodule Questao3 do
  def par(n) do
    rem(n, 2) == 0
  end
end

IO.puts "e par?"
IO.puts Questao3.par(30)
```

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

```
defmodule Questao4 do
        def dobro(n) do
          Enum.map(n, fn (x) -> x * 2 end)
        end
end
IO.inspect Questao4.dobro([1,2,3,4])
```

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

```
defmodule Questao5 do
  def ultimoElemento(n) do
    resultado = Enum.reduce(n, fn(x, y) -> x end)
  end
end
IO.inspect Questao5.ultimoElemento([9.5, 8.0, 7.5, 5])
```

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

```
defmodule Questao6 do
  def fatorial(n) do
    for i <- 0..n, do: multiplicacao(i)
  end
   defp multiplicacao(i), do: i*i
end
IO.inspect Enum.reduce(Questao6.fatorial(10), fn(x, y) -> x end)
```

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

```
defmodule Questao7 do
  def funcao(n,funcao) do
    Enum.map(n,funcao)
  end
end
IO.inspect Questao7.funcao([1, 2, 3, 4], fn(x) -> x + 5 end)
```

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 Questao8 do
  def filtro(n,funcao) do
    Enum.filter(n,funcao)
  end
end
IO.inspect Questao8.filtro([1, 2, 3, 4], fn(x) -> rem(x, 2) == 0 end)
```

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

```
defmodule Questao9 do
def fibonacci(n) do
   for i <- 0..n, do: fb(i)
 end

 defp fb(0), do: 0
 defp fb(1), do: 1
 defp fb(i), do: fb(i-1) + fb(i-2)
end

IO.inspect Questao9.fibonacci(10)
```

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 Questao10 do
 def media(n) do

  apenas_notas = Enum.flat_map(n, fn({n,x}) -> x end)
  resultado = Enum.reduce(apenas_notas, fn(x, acc) -> x + acc end)
  resultado / length(apenas_notas)
 end
end

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

IO.inspect Questao10.media(notas)
```