

UNIVERSIDADE CESUMAR - UNICESUMAR

| **Curso:** Engenharia de Software | | | | | | | | **Série:** 6S | | | | **Turma:** A | | | | | **Turno:** Noite | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Professor(a):** Thiago Bussola da Silva | | | | | | | | **Horário:** | | | | | | | | | | | |
| **Acadêmico (a): JoãoVitor Martins** | | | | | | | | | | | | | | | | **RA: 21011437-2** | | | |
| **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.

R: Funções puras em Elixir, são funções que caracterizam que as variáveis que receberão a mesma entrada de dados, também receberão a mesma saída de dados. Já as funções de ordem superior, são funções na quais permitem passar elas mesmas como parâmetro para outras funções.

funções puras:

sum = fn(a, b) -> a + b end

IO.puts(sum.(2, 5))

ordem superior:

**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.

R: As vantagens da utilização da linguagem Elixir, são: Imutabilidade, fácil de testar, tolerante a falhas. Ou seja, por ter um foco nas funções, que acabam por serem simples, possui: facilidade de entender o código, facilidade de encontrar erros, pois devido a sua composição de funções, torna o código mais previsível.

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

lista = [1, 2, 3, 4, 5]

pares = Enum.filter(lista, fn(x) -> rem(x, 2) == 0 end )

IO.inspect(pares)

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

lista = [1, 2, 3, 4, 5]

dobro = Enum.map(lista, fn(x) -> x \* 2 end)

IO.inspect(dobro)

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

lista = [1, 2, 3, 4]

tam = length(lista)

ultimo = Enum.map(lista, fn(x) -> x[tam])

IO.inspect(ultimo)

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

lista = []

let n = 10

if(filter(x, fn(x) -> rem.(x,2) == 0) end) do

lista = filter(x, fn(x) -> rem.(x,2) == 0) end

else do

:error “Erro”

end

IO.inspect(lista)

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

lista = [1, 2, 3, 4]

funcao = &(&1 + &2)

funcao2 = Enum.map(lista, funcao.(2, 3) end)

IO.inspect(funcao2)

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

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

**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 calcular\_media do**

**fitro\_notas =** Enum.filter(lista, fn(x) -> rem(x, 2) > 0 end )

n = lenght(filtro\_notas)

mapear\_notas = Enum.map(filtro\_notas, fn(x) -> x \* 1 end)

div = Enum.reduce(mapear\_notas, fn(x) => mapear\_notas + acc end)

if(n > 0) {

:ok

results = Enum.reduce(div, fn(x) => div/2 end)

else {

:error ‘erro’

}

IO.inspect(results)