UNIVERSIDADE CESUMAR - UNICESUMAR

Curso: Engenharia de Software	**Série:** 6S	**Turma:** A	**Turno:** Noite
Professor(a): Thiago Bussola da Silva	**Horário:**		
Acadêmico (a): Luiz carlos goncalves de oliveira junior			**RA: 20059362-2**
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 re 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.

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

Uma função pura é uma função que atende a duas características essenciais:

Determinismo: A função sempre produz o mesmo resultado para as mesmas entradas, independentemente do contexto ou estado externo do programa. Isso significa que não há efeitos colaterais envolvidos na execução da função.

Imutabilidade: A função não modifica nenhum dado ou estado fora de seu escopo. Ela não altera variáveis globais, nem realiza operações que afetam o ambiente de execução.

Exemplo: defmodule Exemplo do

```
  def dobro(numero) do
    numero * 2
  end
end

numero = 5
resultado = Exemplo.dobro(numero)
IO.puts("O dobro de #{numero} é #{resultado}") # Saída: O dobro de 5 é 10
```

Funções de ordem superior são funções que podem receber outras funções como argumentos e/ou retornar funções como resultado. Elas tratam as funções como cidadãos de primeira classe, o que significa que as funções são tratadas da mesma forma que qualquer outro valor (como números, strings, etc.).

```
defmodule Exemplo do
  def aplicar_funcao_a_todos(lista, funcao_de_transformacao) do
    Enum.map(lista, funcao_de_transformacao)
  end
end

numeros = [1, 2, 3, 4, 5]
ao_quadrado = Exemplo.aplicar_funcao_a_todos(numeros, fn x -> x * x end)
IO.inspect(ao_quadrado) # Saída: [1, 4, 9, 16, 25]
```

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 uma linguagem funcional escalável, com concorrência eficiente, ótima para sistemas distribuídos e tempo real, graças ao seu ambiente Erlang. É ideal para aplicações de alta concorrência, como telecomunicações, jogos online, chatbots e sistemas críticos tolerantes a falhas.

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

```
defmodule Verificador do
  defpar?(numero) when rem(numero, 2) == 0, do: true
  defpar?(_numero), do: false
end

numero = 10
if Verificador.defpar?(numero) do
  IO.puts("#{numero} é par.")
else
  IO.puts("#{numero} não é par.")
end
```

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

```
defmodule Exemplo do
  def dobro_de_cada_elemento(lista) do
    Enum.map(lista, fn elemento -> elemento * 2 end)
  end
end

numeros = [1, 2, 3, 4, 5]
resultado = Exemplo.dobro_de_cada_elemento(numeros)
IO.inspect(resultado) # Saída: [2, 4, 6, 8, 10]
```

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

```
defmodule Exemplo do
  def ultimo_elemento(lista) do
    lista_invertida = Enum.reverse(lista)
    hd(lista_invertida)
  end
end

minha_lista = [1, 2, 3, 4, 5]
ultimo = Exemplo.ultimo_elemento(minha_lista)
IO.puts("O último elemento da lista é #{ultimo}") # Saída: O último elemento da lista é 5
```

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

```
defmodule Exemplo do
  def calcular_fatorial(0), do: 1
  def calcular_fatorial(n) when n > 0, do: n * calcular_fatorial(n - 1)
end

numero = 5
resultado = Exemplo.calcular_fatorial(numero)
IO.puts("O fatorial de #{numero} é #{resultado}") # Saída: O fatorial de 5 é 120
```

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

```
defmodule Exemplo do
  def aplicar_funcao_a_todos(lista, funcao_de_transformacao) do
    Enum.map(lista, funcao_de_transformacao)
  end
end

numeros = [1, 2, 3, 4, 5]
ao_quadrado = Exemplo.aplicar_funcao_a_todos(numeros, fn x -> x * x end)
IO.inspect(ao_quadrado) # Saída: [1, 4, 9, 16, 25]
```

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 Exemplo do
  def filtrar_lista(lista, funcao_de_filtro) do
    Enum.filter(lista, funcao_de_filtro)
  end
end

numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Defina uma função de filtro que retorna true para números pares
funcao_filtro_par = fn numero -> rem(numero, 2) == 0 end

# Use a função de filtro para obter uma lista de números pares
pares = Exemplo.filtrar_lista(numeros, funcao_filtro_par)
IO.inspect(pares) # Saída: [2, 4, 6, 8, 10]
```

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

```
defmodule Exemplo do
  def sequencia_fibonacci(n) when n <= 0, do: []
  def sequencia_fibonacci(1), do: [0]
  def sequencia_fibonacci(2), do: [0, 1]
  def sequencia_fibonacci(n) do
    sequencia_fibonacci(n, [1, 0])
  end

  defp sequencia_fibonacci(2, acc), do: Enum.reverse(acc)
  defp sequencia_fibonacci(n, [h | t]) do
    sequencia_fibonacci(n - 1, [h + hd(t), h | t])
  end
end
```

```
n = 10
fibonacci = Exemplo.sequencia_fibonacci(n)
IO.inspect(fibonacci) # Saída: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

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