UNIVERSIDADE CESUMAR - UNICESUMAR

Curso: Engenharia de Software	**Série:** 6S	**Turma:** A	**Turno:** Noite
Professor(a): Thiago Bussola da Silva	**Horário:**		
Acadêmico (a): Caetano Fortunato Veiga			**RA: 21170867-2**
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.

Uma função pura é uma função que dados os mesmos argumentos em execuções diferentes, ela irá retornar os mesmos resultados.

Exemplo: function soma(a, b) {return a + b}

Uma função de ordem superior é uma função que aceita outras funções como argumentos dela mesma (e pode retornar funções também).

Exemplo: function conta(funcao(x, y), a, b) {
    return funcao(a, b)
}

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.

A programação em Elixir se destaca por ser em código funcional, e entre suas vantagens estão um código

mais claro, de fácil entendimento, robusto, fácil manutenção, fácil de testar, menos bugs, entre outras.

Devemos utilizá-la em cenários onde a equipe preze por tais características oferecidas pela linguagem,

mesmo podendo tornar o desenvolvimento mais devagar no início.

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

```
def checkPar(number), do: Integer.mod(number, 2)
if Math.checkPar(2) == 0 do
IO.puts "Número par!"
else
IO.puts "Número ímpar!"
end
```

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

```
def doubleList(list), do: Enum.map(list, fn x -> x * 2 end)
lista = [1, 2, 3, 4, 5]
ex4 = Math.doubleList(lista)
IO.inspect(aux)
```

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

```
def lastList(list), do: List.last(list)
ex5 = Math.lastList(lista)
IO.puts(ex5)
```

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

```
def fatorial(1), do: 1
def fatorial(number), do: number * fatorial(number-1)
ex6 = Math.fatorial(5)
IO.puts(ex6)
```

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

```
#incompleto
def double(number), do: number * 2
def doubleLista(list, function()), do: Enum.map(list, fn x -> function(x) end)
ex7 = Math.doubleLista(lista, Math.double())
IO.puts(ex7)
```

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

```
#incompleto
def filterLista(list, function()), do: Enum.map(list, fn x -> function(list, x -> x > 1) end)
ex8 = Math.filterLista(lista, Enum.filter())
```

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

```
def fibo(0), do: 0
def fibo(1), do: 1
def fibo(n), do: n + fibo(n-1)
ex9 = Math.fibo(5)
IO.puts(ex9)
```

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

```
def filtrar(lista), do: Enum.filter(lista, fn {_, notas} -> length(notas) > 0 end)
def somarNotas(lista), do: Enum.reduce(lista, fn(x, acc) -> x + acc end)
notas_validas = LastExercise.filtrar(notas_da_turma)
soma_notas = LastExercise.somarNotas(notas_validas)
IO.puts(soma_notas)
```

Código da prova:

```
defmodule Math do

# ex3
def checkPar(number), do: Integer.mod(number, 2)

#ex4
def doubleList(list), do: Enum.map(list, fn x -> x * 2 end)

#ex5
def lastList(list), do: List.last(list)

#ex6
def fatorial(1), do: 1

def fatorial(number), do: number * fatorial(number-1)

#ex7 - incompleto
def double(number), do: number * 2
```

```
def doubleLista(list, function()), do: Enum.map(list, fn x -> function(x) end)


#ex8 - incompleto
def filterLista(list, function()), do: Enum.map(list, fn x -> function(list, x -> x > 1) end)

#ex9
def fibo(0), do: 0

def fibo(1), do: 1

def fibo(n), do: n + fibo(n-1)

end

if Math.checkPar(2) == 0 do
IO.puts "Número par!"
else
IO.puts "Número ímpar!"
end

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


ex4 = Math.doubleList(lista)
IO.inspect(aux)


ex5 = Math.lastList(lista)
IO.puts(ex5)


ex6 = Math.fatorial(5)
IO.puts(ex6)


ex7 = Math.doubleLista(lista, Math.double())
IO.puts(ex7)


ex8 = Math.filterLista(lista, Enum.filter())
```

```
ex9 = Math.fibo(5)
IO.puts(ex9)

#ex10 - incompleto, não consegui separar os nomes das notas
defmodule LastExercise do

def filtrar(lista), do: Enum.filter(lista, fn {_, notas} -> length(notas) > 0 end)

def somarNotas(lista), do: Enum.reduce(lista, fn(x, acc) -> x + acc 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", []},
]

notas_validas = LastExercise.filtrar(notas_da_turma)

soma_notas = LastExercise.somarNotas(notas_validas)

IO.puts(soma_notas)
```