UNIVERSIDADE CESUMAR - UNICESUMAR

Curso: Engenharia de Software	**Série:** 6S	**Turma:** A	**Turno:** Noite
Professor(a): Thiago Bussola da Silva	**Horário:**		
Acadêmico (a):Luigi siqueira Capoia			**RA:21013423-2**
Disciplina: Paradigmas de Programação			**Data:29/09/2023**

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	

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

A diferença entre essas funções está na composição de uma delas onde as funções puras podem estar contidas dentro das funções de ordem superior, enquanto a função pura não pode conter uma função de ordem superior pois a ordem superior é mais complexa que a pura.

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 linguagem de programação Elixir não é uma linguagem fortemente tipada, tem tolerância a falhas e tem um rápido compilador pode até mesmo ser compilado na web devemos utilizá-la em projetos com baixa complexidade em que temos objetivos simples de resolver. Além de ter uma Escalabilidade podendo ter processos isolados e pode ser utilizado a programação funcional que ajuda os desenvolvedores a escrever códigos curtos, concisos e sustentáveis.

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

```
lista =[1,2]
 verifica = Enum.map(lista, fn n -> if rem(n, 2) == 0 do "par "
     else
       "ímpar "
     end
   end)
IO.puts(verifica)
```

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

```
lista =[1,2,3,4,5,6]
multiplicando=Enum.map(lista,fn item -> item * 2 end )
IO.puts(multiplicando)
```

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

```
lista =[3,80,9,2,8,50]
 ultimo =   Enum.reduce(lista, fn x, acc -> if x !== acc, do: x, else: acc end)

IO.puts(ultimo)
```

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

```
defmodule Fatorial do
  def fatorial(n), do: fatorial(n * (n-1)) end
IO.puts(Fatorial.fatorial(2))
```

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

```
lista =[1,2,3,4,5,6]
multiplicando=Enum.map(lista,fn item -> item * 2 end )

 verifica = Enum.map(multiplicando, fn n -> if rem(n, 2) == 0 do "par "
     else
       "ímpar "
     end
   end)
IO.puts(verifica)
```

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

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

defmodule Exercicio10 do
  def media(notas_da_turma) do
    resgistros = Enum.filter (notas_da_turma , fn{nome, notas} -> length (notas) > 0
end)
```

```
  IO.puts(resgistros)
   end
  end
```

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

```
defmodule Fibonacci do
   def fibonacci(0), do: 0
   def fibonacci(1), do: 1
   def fibonacci(n), do: fibonacci(n-1) + fibonacci(n-2)
  end
  IO.puts(Fibonacci.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", []},]
defmodule Exercicio10 do

  def media(notas_da_turma) do

   resgistros = Enum.filter (notas_da_turma , fn{nome, notas} -> length (notas) > 0
end)

   todas_as_notas = Enum.map (registros_validos, fn {nomes, notas} -> notas end)

    {soma, quantidade} = Enum.reduce(todas_as_notas, {0.0, 0}, fn nota, {soma,
quantidade} ->
     if nota > 0.0 do
       {soma + nota, quantidade+ 1}
     else
       {soma, quantidade}
```

```
      end
     end)

    if quantidade > 0 do
      media = soma / quantidade
      IO.puts("A media da turma é #{media}")
    else
      IO.puts("Nao foi possivel calcular a media da turma")
    end
  end
end
```