Data: 22/08/2024_____

Nome: Randerson Douglas Ribeiro dos Santos_____

E-mail: randersondouglas.r@gmail.com_____

Github: https://github.com/rdouglas10_____

Início do teste: _____:_____ **Término do teste: _____:_____**

Prova v20200424 – Tempo total (1 hora e 30 minutos)

Parte I – Questões de múltipla escolha.

	1	2	3	4	5
a					
b					
c					
d					
e					

Parte II – Algoritmos

Parte III – Questão dissertativa

Parte IV – Questões sobre você

Parte I – Questões de múltipla escolha. (sugestão: 30 minutos)

1) Avalie o seguinte código:

```
int fn(int v)
{
if(v==1 || v==0)
return 1;
if(v%2==0)
6.      return fn(v/2)+2;
7. else
8.      return fn(v-1)+3;
9. }
```

Qual o resultado esperado de **`fn(7)`**?

a) 10
b) 11 X
c) 1
d) 9
e) 7

2) Considere as seguintes afirmações sobre expressões regulares:

I. O padrão **`a{3}b?c*`** ocorre na sequência de caracteres **`aaab`**.
II. O padrão **`a*b`** ocorre na sequência de caracteres **`b`**.
III. O padrão **`a{1,3}b?c*`** ocorre na sequência de caracteres **`aaab`**.

Analisando as afirmações se conclui que:

a) **Apenas as afirmações I e II são verdadeiras. X**
b) Apenas as afirmações II e III são verdadeiras.
c) Apenas a afirmação I é verdadeira.
d) Apenas as afirmações I e III são verdadeiras.
e) Todas as afirmações são verdadeiras.

3) Qual das seguintes estruturas de dados é, em média, a mais rápida para recuperar um item de dados escolhido aleatoriamente?

a) Binary Tree
b) Stack
c) Hash Table X
d) Queue
e) Linked List

4) Considere um algoritmo que possa ser solucionado em diferentes ordens de complexidade. Qual das opções abaixo representa a correta ordenação do mais eficiente (mais a esquerda) para o menos eficiente (mais a direita):

a) O(1) ; O(n) ; O(n log n) ; O(log n) ; O(n^2) ; O(n!); O(2^n)
b) O(1) ; O(log n) ; O(n) ; O(n log n) ; O(n^2) ; O(2^n) ; O(n!) X
c) O(n!) ; O(2^n) ; O(n^2) ; O(n log n) ; O(n) ; O(log n) ; O(1)
d) O(1) ; O(n) ; O(n log n) ; O(log n) ; O(n^n) ; O(2^n) ; O(n!)
e) O(n!) ; O(2^n) ; O(n^2) ; O(log n) ; O(n log n) ; O(n) ; O(1)

5) Considere as tabelas a seguir, criadas em um banco de dados relacional através da linguagem SQL.

```
1) CREATE TABLE Funcionario
2)  ( fcod int PRIMARY KEY,
3)    nome varchar (32),
4)    salario number (7,2),
5)    dcod int FOREIGN KEY REFERENCES Diretoria (dcod));
6) CREATE TABLE Diretoria
7)  ( dcod int PRIMARY KEY,
8)    dnome varchar (12),
9)    chefe int FOREIGN KEY REFERENCES Funcionario (fcod));
```

Sejam as consultas (C1, C2 e C3) também em SQL, a seguir.

```
C1. SELECT nome, salario FROM Funcionario F, Diretoria D
WHERE F.dcod = D.dcod AND F.fcod = D.chefe;
C2. SELECT nome, salario FROM Funcionario as F INNER JOIN
Diretoria as D ON F.dcod = D.dcod WHERE F.fcod = D.chefe;
C3. SELECT nome, salario FROM F.fcod = D.chefe;
```

a) Apenas a consulta C1 retorna o nome e o salário dos chefes das diretorias.
b) Apenas a consulta C2 retorna o nome e o salário dos chefes das diretorias.
c) Apenas a consulta C3 retorna o nome e o salário dos chefes das diretorias.
d) As consultas C1, C2 e C3 são equivalentes e retornam o nome e o salário dos chefes das diretorias.
e) As consultas C1 e C2 são equivalentes e retornam o nome e o salário dos chefes das diretorias.

Parte II – Algoritmos – resolver as duas questões

1) Desenvolva um algoritmo que, para cada número inteiro dentro do intervalo fechado entre 0 e 16, imprima:
   - "foo" se o número for divisível por 3
   - "bar" se o número for divisível por 5
   - "baz" se o número for divisível por 3 e por 5
   - o próprio número caso contrário

Resposta:

```
# Itera sobre cada número no intervalo fechado entre 0 e 16
for num in range(17):
  if num % 3 == 0 and num % 5 == 0:
     print("baz")  # Divisível por 3 e 5
  elif num % 3 == 0:
     print("foo")  # Divisível por 3
  elif num % 5 == 0:
      print("bar")  # Divisível por 5
  else:
      print(num)  # Caso contrário, imprime o próprio número
```

2) Implemente um algoritmo que receba como entrada uma sequência de Strings – que podem ou não apresentar repetições – e imprima, uma única vez para cada termo, a quantidade de vezes em que o mesmo foi encontrado.

   Obs: a formatação da saída é livre.

   Ex:

   Entrada: **["PaTiNeTe", "SKATE", "Patinete", "BicicletA"]**

   Saída:

```
skate=1
patinete=2
bicicleta=1
```

Resposta:

```
def contar_ocorrencias(strings):
    # Converte todas as strings para minúsculas
    strings = [s.lower() for s in strings]

    # Usa o Counter para contar a frequência de cada string
    contagem = Counter(strings)

    # Imprime cada termo e sua quantidade de ocorrências
    for termo, quantidade in contagem.items():
        print(f"{termo}={quantidade}")

# Exemplo de uso
entrada = ["PaTiNeTe", "SKATE", "Patinete", "BicicletA"]
contar_ocorrencias(entrada)
```

Parte III – Questão de arquitetura – escolha um item

Faça um diagrama de arquitetura de sistemas que represente **apenas um** dos casos de uso abaixo:

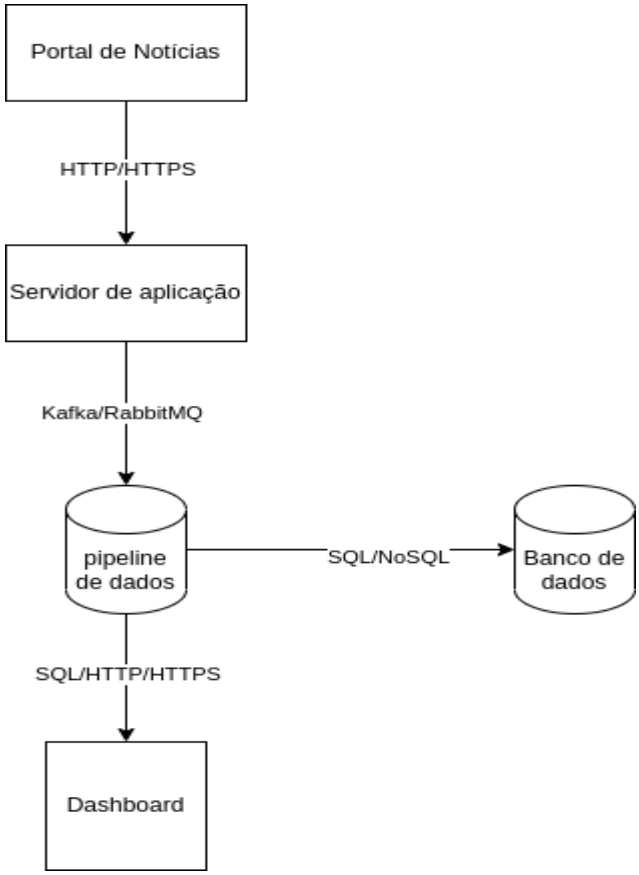
a. Efetuar uma compra em um site de e-commerce com a opção de boleto.
b. Gerar um dashboard de número de acessos - em tempo real - por categoria (ex: política, esporte, economia) feitos em um portal de notícias.

Requisitos:

- Especificar os componentes e sistemas externos em alto nível, indicando a funcionalidade de cada um e a maneira (protocolo) como comunicam entre si.
- Indicar e justificar a escolha de infraestrutura da solução (cloud, on premise ou mista).

Obs: caso esteja respondendo a prova no computador, utilize a ferramenta https://draw.io

Segundo caso (Letra B)



Parte IV – Questões sobre você

1) Como você adquire novos conhecimentos? Quais suas fontes de estudo?

Busco participar de fóruns e eventos, para compartilhar informações e ver diferentes pontos de vista e comparar com o que existe na literatura, afim de agregar mais conhecimento e otimizar o trabalho.

2) Qual livro técnico você recomendaria a alguém? Por quê?

Design patterns, clean architecture e clean code. Juntos, podem agregar bastante conhecimento, no quesito de estruturação, boas práticas e aplicação de padrões que permitem que o código se mantenha escalável.