TRABALHO FINAL (TDE)

Este trabalho final consiste na elaboração de um jogo da forca em python, cujo tema central das perguntas deve ser Lógica matemática aplicada a conjuntos. Os critérios avaliativos e descritivos para o projeto são:

a) O grupo de trabalho poderá ser composto de um grupo de até 5 alunos.(inclusive o trabalho pode ser individual).
b) Os alunos irão entregar um artigo científico sobre o tema.
c) Objetivo central do trabalho será desenvolver um jogo da forca em python que trabalhe o conceito de conjuntos e lógica matemática.
d) Em anexo ao artigo científico deve ser colocado imagens ilustrativas do jogo e o algoritmo desenvolvido.
e) Será permitido uso de inteligência artificial, bibliotecas, livros e demais fontes de desenvolvimento.
f) Os critérios avaliativos serão:

- Complexidade, eficiência e qualidade do algoritmo desenvolvido. (0.5 pontos)
- Criatividade e inovação no desenvolvimento do jogo. (0.5 pontos)
- Assertividade, coesão e veracidade das perguntas e respostas inseridas (0.5 pontos)
- Elaboração do artigo dentro das normas acadêmicas. (0.5 pontos)

Em sala de aula:

- Estude o modelo de artigo científico(em anexo) e tire suas dúvidas com o professor.
- Leia o exemplo de código do projeto em anexo.
- Assista o vídeo indicado: https://www.youtube.com/watch?v=fcPSU5t3CIo
- Execute inicialmente o código comentado.
- Elabore 10 perguntas envolvendo o tema de trabalho e seu gabarito, usando livros, conceitos aprendidos, sites, inteligência artificial, entre outros meios de desenvolvimento.
- Discuta com seus colegas como podemos implementar este jogo para trabalhar conceitos de lógica matemática aplicada a conjuntos.
- Como modificar o algoritmo para executar essa tarefa?
- Que tipo de melhorias deixariam o Jogo mais atrativo?
- Após a finalização do projeto, mostre tanto o projeto, quanto o artigo para análise do professor, logo após entregue o trabalho no ava.

Anexo 1: modelo de algoritmo

```
import pygame as pg  # Importa a biblioteca Pygame e a renomeia como 'pg' para facilitar o uso
import random  # Importa a biblioteca random, que permite gerar números aleatórios

# Cores do jogo definidas em formato RGB (Red, Green, Blue)
branco = (255, 255, 255)  # Branco puro
preto = (0, 0, 0)  # Preto puro

# Setup da tela do Jogo com dimensões 1000x600
window = pg.display.set_mode((1000, 600))  # Cria a janela do jogo com largura de 1000 pixels e altura de 600 pixels

# Inicializando o sistema de fontes do Pygame para exibir textos
pg.font.init()  # Inicializa as funções de fontes no Pygame

# Escolhendo a fonte "Courier New" com tamanho 50 para exibir a palavra camuflada no jogo
fonte = pg.font.SysFont("Courier New", 50)
# Escolhendo a mesma fonte, porém com tamanho 30, para o botão "Restart"
fonte_rb = pg.font.SysFont("Courier New", 30)

# Lista de palavras que podem ser sorteadas aleatoriamente no jogo da forca
palavras = ['PARALELEPIPEDO', 'ORNITORINCO', 'APARTAMENTO', 'XICARA DE CHA']

# Variáveis iniciais
tentativas_de_letras = [' ', '-']  # Lista que contém as letras já tentadas, iniciando com espaço e hífen
palavra_escolhida = ''  # Armazena a palavra sorteada, inicialmente vazia
palavra_camuflada = ''  # Armazena a palavra camuflada (com letras ocultas), inicialmente vazia
end_game = True  # Indica se o jogo terminou ou não, inicia como verdadeiro para sortear uma nova palavra
chance = 0  # Contador de chances/erros do jogador
letra = ' '  # Armazena a última letra tentada, inicialmente vazia
click_last_status = False  # Armazena o estado anterior do clique do mouse

# Função para desenhar a forca e partes do corpo conforme o número de erros (chance)
def Desenho_da_Forca(window, chance):
    # Preenche o fundo da tela com a cor branca
    pg.draw.rect(window, branco, (0, 0, 1000, 600))
    # Desenha a base e a estrutura da forca
    pg.draw.line(window, preto, (100, 500), (100, 100), 10)  # Linha vertical da base
    pg.draw.line(window, preto, (50, 500), (150, 500), 10)   # Linha horizontal da base
```

```
    pg.draw.line(window, preto, (100, 100), (300, 100), 10)  # Linha horizontal do topo
    pg.draw.line(window, preto, (300, 100), (300, 150), 10)  # Linha vertical do topo
    # Desenho das partes do corpo conforme o número de erros
    if chance >= 1:  # Cabeça é desenhada se o jogador erra 1 vez ou mais
        pg.draw.circle(window, preto, (300, 200), 50, 10)  # Desenha a cabeça
    if chance >= 2:  # Tronco é desenhado se o jogador erra 2 vezes ou mais
        pg.draw.line(window, preto, (300, 250), (300, 350), 10)  # Desenha o tronco
    if chance >= 3:  # Braço direito é desenhado se o jogador erra 3 vezes ou mais
        pg.draw.line(window, preto, (300, 260), (225, 350), 10)  # Desenha o braço direito
    if chance >= 4:  # Braço esquerdo é desenhado se o jogador erra 4 vezes ou mais
        pg.draw.line(window, preto, (300, 260), (375, 350), 10)  # Desenha o braço esquerdo
    if chance >= 5:  # Perna direita é desenhada se o jogador erra 5 vezes ou mais
        pg.draw.line(window, preto, (300, 350), (375, 450), 10)  # Desenha a perna direita
    if chance >= 6:  # Perna esquerda é desenhada se o jogador erra 6 vezes ou mais (fim de
jogo)
        pg.draw.line(window, preto, (300, 350), (225, 450), 10)  # Desenha a perna esquerda

# Função para desenhar o botão de "Restart" na tela
def Desenho_Restart_Button(window):
    pg.draw.rect(window, preto, (700, 100, 200, 65))  # Desenha um retângulo preto para o
botão
    texto = fonte_rb.render('Restart', 1, branco)  # Renderiza o texto "Restart" na cor branca
    window.blit(texto, (740, 120))  # Exibe o texto na posição especificada dentro do botão

# Função para sortear uma nova palavra quando o jogo começa ou reinicia
def Sorteando_Palavra(palavras, palavra_escolhida, end_game):
    if end_game == True:  # Se o jogo terminou (ou está reiniciando)
        palavra_n = random.randint(0, len(palavras) - 1)  # Escolhe um índice aleatório da lista
de palavras
        palavra_escolhida = palavras[palavra_n]  # Seleciona a palavra correspondente ao
índice sorteado
        end_game = False  # Sinaliza que o jogo está em andamento
        chance = 0  # Reinicia o contador de erros
    return palavra_escolhida, end_game  # Retorna a palavra sorteada e o estado do jogo

# Função para camuflar a palavra sorteada, ocultando as letras não adivinhadas
def Camuflando_Palavra(palavra_escolhida, palavra_camuflada, tentativas_de_letras):
    palavra_camuflada = palavra_escolhida  # Começa com a palavra inteira
    for n in range(len(palavra_camuflada)):  # Percorre cada letra da palavra
        if palavra_camuflada[n:n + 1] not in tentativas_de_letras:  # Se a letra não foi tentada
ainda
            palavra_camuflada = palavra_camuflada.replace(palavra_camuflada[n], '#')  #
Substitui a letra por '#'
    return palavra_camuflada  # Retorna a palavra camuflada

# Função para processar uma nova tentativa de letra
def Tentando_uma_Letra(tentativas_de_letras, palavra_escolhida, letra, chance):
    if letra not in tentativas_de_letras:  # Se a letra ainda não foi tentada
```

```
        tentativas_de_letras.append(letra)  # Adiciona a letra à lista de tentativas
        if letra not in palavra_escolhida:  # Se a letra não está na palavra
            chance += 1  # Incrementa o contador de erros (chance)
    elif letra in tentativas_de_letras:  # Se a letra já foi tentada, não faz nada
        pass
    return tentativas_de_letras, chance  # Retorna as tentativas e o número de erros

# Função para exibir a palavra (camuflada ou completa) na tela
def Palavra_do_Jogo(window, palavra_camuflada):
    palavra = fonte.render(palavra_camuflada, 1, preto)  # Renderiza a palavra camuflada
usando a fonte
    window.blit(palavra, (200, 500))  # Exibe a palavra na tela na posição (200, 500)

# Função para reiniciar o jogo se o botão de restart for clicado
def Restart_do_Jogo(palavra_camuflada, end_game, chance, letra, tentativas_de_letras,
click_last_status, click, x, y):
    count = 0  # Inicializa o contador de letras adivinhadas
    limite = len(palavra_camuflada)  # Define o limite como o comprimento da palavra
    for n in range(len(palavra_camuflada)):  # Percorre a palavra camuflada
        if palavra_camuflada[n] != '#':  # Conta as letras já reveladas
            count += 1
    if count == limite and click_last_status == False and click[0] == True:  # Se todas as letras
foram reveladas e o botão foi clicado
        if x >= 700 and x <= 900 and y >= 100 and y <= 165:  # Verifica se o clique foi na área
do botão de restart
            tentativas_de_letras = [' ', '-']  # Reseta as tentativas de letras
            end_game = True  # Marca o fim do jogo para iniciar um novo
            chance = 0  # Reseta o contador de chances
            letra = ' '  # Reseta a última letra
    return end_game, chance, tentativas_de_letras, letra  # Retorna o estado do jogo
atualizado

# Loop principal do jogo
while True:
    # Verifica os eventos que estão acontecendo no jogo (teclado, mouse, etc.)
    for event in pg.event.get():
        if event.type == pg.QUIT:  # Se o jogador fechar a janela do jogo
            pg.quit()  # Encerra o Pygame
    # Lê a posição do mouse e o status dos cliques
    x, y = pg.mouse.get_pos()
    click = pg.mouse.get_pressed()

    # Desenho da forca e botão de restart
    Desenho_da_Forca(window, chance)
    Desenho_Restart_Button(window)

    # Sorteia uma nova palavra se o jogo começou/reiniciou
```

```
    palavra_escolhida, end_game = Sorteando_Palavra(palavras, palavra_escolhida,
end_game)

    # Atualiza a palavra camuflada conforme as tentativas
    palavra_camuflada = Camuflando_Palavra(palavra_escolhida, palavra_camuflada,
tentativas_de_letras)

    # Exibe a palavra camuflada na tela
    Palavra_do_Jogo(window, palavra_camuflada)

    # Reinicia o jogo se o botão de restart for clicado
    end_game, chance, tentativas_de_letras, letra = Restart_do_Jogo(palavra_camuflada,
end_game, chance, letra, tentativas_de_letras, click_last_status, click, x, y)

    # Atualiza o status do último clique do mouse
    click_last_status = click[0]

    # Atualiza a tela do jogo
    pg.display.update()
```

Anexo 2: Modelo de artigo científico

TÍTULO DO ARTIGO:

SUBTÍTULO

Nome e sobrenome do primeiro autor[1*]
Nome e sobrenome do segundo autor[2**]

RESUMO

Introduzem-se os conceitos de Informação e de Gestão da informação no âmbito da Ciência da Informação e o seu valor operativo quando aplicado a Sistemas de Informação com um elevado nível de complexidade. Desenvolve-se o conceito de Sistema de Informação e o pensamento sistémico a partir dos estudos da Teoria Geral dos Sistemas de Ludwig von Bertalanffy e da relação estabelecida por Piero Mella entre estrutura ou unidade e sistema de informação. Abordam-se, genericamente, as possíveis classificações e tipologias de sistemas. Analisa-se o conceito de Sistema Tecnológico de Informação e as suas relações com a Gestão da Informação das Organizações. Conclui-se com a análise das implicações e dos desafios da Gestão do Conhecimento na criação da Inteligência Competitiva e da Gestão da Inovação nas organizações. Exemplo de resumo retirado de Marques (2017).

Palavras-chave: palavra 1; palavra 2; palavra 3; palavra 4.

ABSTRACT

Resumo em outro idioma. Elemento opcional.

Keywords: keyword 1; keyword 2; keyword 3; keyword 4.

1 INTRODUÇÃO

"A introdução é a parte inicial do artigo na qual devem constar a delimitação do assunto tratado, os objetivos da pesquisa e outros elementos necessários para situar o tema do artigo." (Associação Brasileira de Normas Técnicas, 2018, p. 5).

> Exemplo de citação direta - A citação direta, com mais de três linhas, deve ser destacada com recuo padronizado em relação à margem esquerda, com letra menor

[1*] Pequeno currículo dos autores, contendo a vinculação institucional e endereço de e-mail. Para artigos entregues em disciplinas de cursos, este espaço pode ser utilizado para informações institucionais como o nome da Universidade, do curso, da disciplina e do professor responsável.

[2**] Profissão – Instituição a que está vinculado. Titulação. E-mail: xxx@xxx.com.br.

> que a utilizada no texto, em espaço simples e sem aspas. Recomenda-se o recuo de 4 cm. (Associação Brasileira de Normas Técnicas, 2023, p. 12).

2 TÍTULO DO CAPÍTULO

"Desenvolvimento é a parte principal do artigo, que contém a exposição ordenada e pormenorizada do assunto tratado. Divide-se em seções e subseções, conforme ABNT NBR 6024." (Associação Brasileira de Normas Técnicas, 2018, p. 5).

2.1 Exemplo de ilustrações e tabelas

As figuras devem ser apresentadas conforme exemplo da

Figura 1 – Biblioteca UFFS Campus Chapecó

Fonte: Simioni (2017).

A formatação das tabelas deve seguir a Norma de Formatação Tabular do IBGE, que está disponível no link: https://biblioteca.ibge.gov.br/visualizacao/livros/liv23907.pdf A Tabela 1 é um exemplo de como deve ser apresentada uma tabela em um trabalho acadêmico:

Tabela 1 – Variação IGPM

Mês/Ano	**%**
07/2020	0,49
08/2020	0,53
09/2020	0,82

10/2020	0,65

Fonte: Calcular [...] (2020).

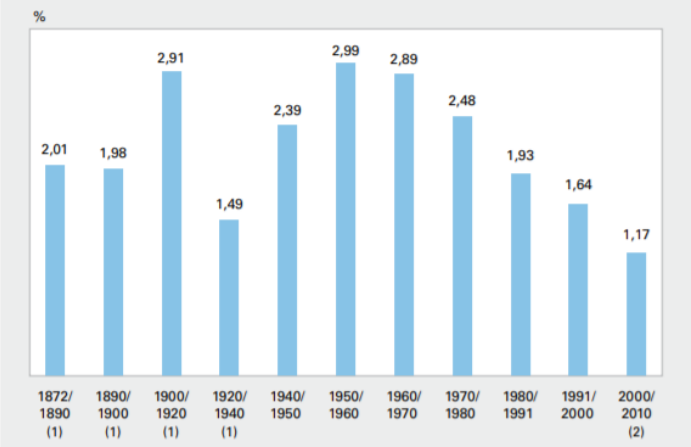
A principal diferença entre um quadro e uma tabela é o seu conteúdo: uma tabela contém números e um quadro contém texto. O Quadro 1 mostra como um quadro deve ser apresentado em um trabalho acadêmico. As normas da ABNT não informam sobre a formatação dentro do quadro, ficando ela a critério estético do autor. Ela versa apenas sobre o título e a fonte, que devem seguir o exemplo mostrado.

Quadro 1 – Ciclo PDCA

ETAPAS	**AÇÕES**
P (PLAN)	**Planejar o trabalho a ser realizado** por meio de um plano de ação após a identificação, reconhecimento das características e descoberta das causas principais do problema (projeto da garantia da qualidade).
D (DO)	**Realizar o trabalho planejado** de acordo com o plano de ação (execução da garantia da qualidade, cumprimento dos padrões).
C (CHECK)	**Medir ou avaliar** o que foi feito, identificando a diferença entre o realizado e o que foi planejado no plano de ação (verificação do cumprimento dos padrões da qualidade).
A (ACT)	**Atuar corretivamente** sobre a diferença identificada (caso houver); caso contrário, haverá a **padronização** e a conclusão do plano (ações corretivas sobre os processos de planejamento, execução e auditoria; eliminação definitiva das causas, revisão das atividades e planejamento.

Fonte: Adaptado de Chiavenato (2004).

Gráfico 1 – Taxa média geométrica de crescimento anual, Brasil – 1872/2010



Fonte: IBGE (2011).

3 CONSIDERAÇÕES FINAIS

"Considerações finais é a parte final do artigo, na qual se apresentam as considerações correspondentes aos objetivos e/ou hipóteses." (Associação Brasileira de Normas Técnicas, 2018, p. 5).

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 10520**: informação e documentação: citações em documentos: apresentação. 2. ed. Rio de Janeiro: ABNT, 2023.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6022**: informação e documentação: artigo em publicação periódica técnica e/ou científica: apresentação. Rio de Janeiro: ABNT, 2018.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **ABNT NBR 6023**: informação e documentação: referências: elaboração. Rio de Janeiro: ABNT, 2018.

CALCULAR correção monetária IPC do IGP (FGV). *[S.l.]*, 2020. Disponível em: https://www.ecalculos.com.br/utilitarios/ipc-do-igp-fgv.php. Acesso em: 13 nov. 2020.

CHIAVENATO, Idalberto. **Introdução à teoria geral da administração**. 3. ed. rev. e atual. Rio de Janeiro: Elsevier: Campus, 2004.

IBGE. **Sinopse do censo demográfico 2010**. Rio de Janeiro, 2011. Disponível em: https://biblioteca.ibge.gov.br/visualizacao/livros/liv49230.pdf. Acesso em: 16 nov. 2020.

MARQUES, Maria Beatriz. Gestão da informação em sistemas de informação complexos. **Pesquisa Brasileira em Ciência da Informação e Biblioteconomia**, João Pessoa, v. 12, n. 2, p. 60-76, 2017. Disponível em: https://periodicos.ufpb.br/ojs/index.php/pbcib/article/view/35505. Acesso em: 16 jun. 2021.

SIMIONI, Lilian. **Biblioteca reabre para atendimentos depois do inventário anual**. 2017. Disponível em: https://www.uffs.edu.br/campi/chapeco/noticias/imagens/biblioteca-reabre-para-atendimentos-depois-do-inventario-anual-foto-lilian-simioni-arquivo-uffs/@@images/image. Acesso em: 13 nov. 2020.

APÊNDICE A – Título

[Inserir apêndice, se houver].

ANEXO A – Título (elemento opcional)

[Inserir anexo, se houver].

AGRADECIMENTOS

Texto em que o autor faz agradecimentos dirigidos àqueles que contribuíram de maneira relevante à elaboração do artigo.

Modelo de questões

Conjunto A: Conjunto de pessoas que gostam de música clássica.
$A = \{\text{Ana}, \text{Bruno}, \text{Carla}, \text{Daniel}\}$

Conjunto B: Conjunto de pessoas que gostam de jazz.
$B = \{\text{Bruno}, \text{Eduarda}, \text{Fábio}, \text{Gabriela}\}$

Conjunto C: Conjunto de pessoas que gostam de rock.
$C = \{\text{Ana}, \text{Gabriela}, \text{Heitor}, \text{Isabela}\}$

Conjunto D: Conjunto de pessoas que gostam de música eletrônica.
$D = \{\text{Carla}, \text{Eduarda}, \text{Isabela}, \text{João}\}$

1. **Quem gosta de música clássica e jazz ao mesmo tempo (interseção de A e B)?**
   Resposta: Bruno.
2. **Quem gosta de música clássica ou jazz (união de A e B)?**
   Resposta: Ana.
3. **Quem gosta apenas de música clássica, mas não de jazz (diferença de A e B)?**
   Resposta: Daniel.
4. **Quem gosta de rock e não gosta de música eletrônica (diferença de C e D)?**
   Resposta: Heitor.
5. **Quem gosta de música eletrônica ou jazz, mas não de música clássica (união de B e D excluindo A)?**
   Resposta: Eduarda.