Memorial Descritivo – Tabela Verdade

Disciplina: Matemática Discreta
Aluno(s): Raphael Antonio dos Santos Rangel
Data: [Data de Entrega]
Sistema Operacional: Ubuntu 22.04
Linguagem: Python 3
Bibliotecas: `Tkinter`, `Pandas`, `Itertools`

1. Descrição da Atividade

A presente atividade teve como objetivo o desenvolvimento de um sistema interativo voltado à criação e análise de tabelas verdade, com base em conceitos de lógica proposicional e representações visuais.

O sistema desenvolvido permite ao usuário gerar tabelas verdade para Fórmulas Bem Formuladas (FBFs), visualizar subfórmulas e classificar as fórmulas como tautologias, contradições ou contingências. Adicionalmente, a aplicação oferece a funcionalidade de leitura de FBFs a partir de arquivos de texto.

A interface gráfica foi implementada utilizando a biblioteca *Tkinter*, enquanto a lógica de funcionamento foi desenvolvida em linguagem *Python*, seguindo o padrão arquitetural *Model-View-Controller* (MVC). O Modelo é responsável pela lógica de processamento e manipulação das fórmulas, a Visão cuida da interface gráfica do usuário, e o Controlador gerencia a interação entre o Modelo e a Visão, incluindo o tratamento de arquivos.

2. Materiais Consultados

- Documentação oficial do Python
- Tutorial Tkinter – GeeksForGeeks
- Documentação da biblioteca Pandas
- Stack Overflow
- Documentação do Itertools
- ChatGPT & Claude

3. Estrutura do Projeto

- **Representação de Fórmulas Lógicas**: Implementação de estruturas de dados para representar fórmulas bem formadas (FBFs) da lógica proposicional, utilizando classes como **LogicalOperations** para operações básicas e **Parser** para análise sintática.
- **Processamento de Expressões**: Funcionalidades para processar expressões lógicas através da classe **ExpressionProcessor** , que extrai variáveis e tokeniza as expressões para posterior análise.

- **Manipulação de Fórmulas**:A classe **FormulaHandler** gerencia a identificação e manipulação de subfórmulas, permitindo a decomposição de expressões complexas em componentes mais simples.
- **Geração de Tabelas Verdade**: Implementação de algoritmos na classe **TruthTableGenerator** para criar tabelas verdade completas a partir de fórmulas lógicas, avaliando todas as combinações possíveis de valores verdade.
- **Classificação de Fórmulas**: Funcionalidade para classificar automaticamente as fórmulas como tautologias, contradições ou contingências com base nos resultados da tabela verdade.
- **Gerenciamento de Arquivos**: A classe **FileHandler** no componente Controller permite carregar fórmulas de arquivos de texto organizados por categorias na pasta FBF.
- **Interface Gráfica**: Interface completa desenvolvida com *Tkinter* na classe **TruthTableGUI** , incluindo campos de entrada, botões de operação, visualização de tabelas verdade e opções para carregar fórmulas de arquivos.
- **Arquitetura MVC**: O projeto segue o padrão *Model-View-Controller*, com clara separação entre a lógica de negócio (Model), a interface do usuário (View) e o controle de fluxo da aplicação (Controller).
- **Organização de Dados**: Estrutura de pastas bem definida para armazenar diferentes tipos de fórmulas lógicas (tautologias, contradições, contingências, etc.) na pasta FBF.

4. Funcionalidades Implementadas

- **Entrada de fórmulas**: Inserção manual de fórmulas lógicas ou seleção a partir de arquivos pré-definidos.
- **Visualização**: Exibição clara e organizada das tabelas verdade com todas as combinações de valores.
- **Subfórmulas**: Identificação e avaliação automática de todas as subfórmulas componentes da expressão principal.
- **Classificação de fórmulas**: Análise e classificação automática das fórmulas como tautologias, contradições ou contingências.
- **Operadores lógicos**: Suporte completo para operadores AND, OR, NOT, XOR, equivalência e implicação, juntamente com um menu de escolha de operadores.
- **Biblioteca de exemplos**: Conjunto organizado de fórmulas em diferentes categorias para testes e aprendizado.

5. Uso de Inteligência Artificial

A IA (ChatGPT e Claude) foi usada para:

- Estudo de técnicas para o desenvolvimento de interfaces gráficas em *Python* por meio da biblioteca *Tkinter*.
- Sugeriu aprimoramentos na estrutura modular do código, promovendo melhor organização e legibilidade;
- Auxiliou na verificação do código e na documentação do mesmo;
- Indicou boas práticas de engenharia de software, uso de funções com documentação e tratamento de exceções.

6. Estratégia de Aprendizagem

Utilizei de videoaulas do professor Hemerson Pistori para entendimento inicial da matéria e do uso de inteligência artificiais para a realização de um esboço inicial e de ajustes finais. As dúvidas foram resolvidas com a ajuda de colegas e professores em sala de aula e do uso do Stack Overflow.

7. Tempo Total Gasto (aproximado)

Etapa	**Tempo Estimado**
Estudo da lógica de resolução	4 horas
Implementação da lógica	8 horas
Construção da interface gráfica	6 horas
Testes e ajustes finais	3 horas
Elaboração do relatório	2 horas
Total aproximado	**23 horas**

8. Boas Práticas de Engenharia de Software

- Padrão MVC (Model-View-Controller)
- Separação clara entra a interface do usuário e a lógica do programa
- Uso de funções bem definidas com `docstrings`.
- Tratamento de exceções com `try-except`.
- Validação de entradas via interface.
- Uso de bibliotecas consagradas e código limpo e comentado.
- Programação Orientada a Objetos.

9. Link para o Vídeo Demonstrativo

Link: [Inserir link do vídeo no YouTube]
(o vídeo será adicionado posteriormente demonstrando a execução no Ubuntu 22.04)