MODUL UNIVERSITY
PRIVATE UNIVERSITY VIENNA

Name: _____

Semester Project: Complex Algorithm Analysis and Data Sorting Tool Using Truth Tables (Console-based application)

Overview

In this project, you will be developing a tool that processes complex data objects and evaluates the efficiency of various sorting and searching algorithms. These data objects will have both numerical and logical properties, and your task is to implement sorting that incorporates truth tables and logical propositions. You'll be working with object-oriented programming principles while analyzing the performance of your algorithms using loops, recursion, and a variety of data types.

You have the freedom to choose any dataset that interests you. Whether it's related to e-commerce, scheduling, or something entirely different, your goal is to build a system around that dataset, applying appropriate algorithms to solve a specific problem. The system could involve sorting products, organizing a schedule, or analyzing complex logical expressions—it's up to you! Some examples recommended by ChatGPT are:

- Student Management System
- E-commerce Inventory Management System
- Task Management System with Time Complexity Analysis
- Library Management System
- Recipe Recommendation System
- Pathfinding in a Maze (Game)
- Social Media Post Analysis System
- Budget Tracking and Financial Management System
- Hospital Patient Management System
- Travel Route Planning System
- E-Learning Platform with Course Recommendation
- Job Matching and Recruitment System
- Fitness Tracking Application
- Virtual Event Management System
- Etc.

Make sure to be creative and innovative.

Performance analysis will be an important part of your project, so make sure to assess how your algorithms perform as dataset size and logical complexity increase, and understand how recursion and truth table evaluations impact efficiency.

MODUL UNIVERSITY
PRIVATE UNIVERSITY VIENNA

Name: _____

Key Components

1. **Sorting Algorithms**: Implement at least **1 sorting algorithms** (Insertion sort, Merge sort, Selection sort, etc.), with the following requirements:
    - At least **one sorting algorithm must use a loop** (e.g., Insertion Sort, Bubble Sort).
    - At least **one sorting algorithm must use recursion** (e.g., Merge Sort, Recursive Selection Sort).
    - Sorting is based on two layers:
        - **Primary Sorting**: Sort based on numerical, string, or float data properties.
        - **Secondary Sorting**: Use logical propositions and **truth tables** as secondary criteria. For example, sort objects where logical expressions ($p \wedge q$, $p \rightarrow q$) evaluate to True before those that evaluate to False.
2. **Truth Table and Logical Complexity**:
    - Each object will have an associated logical expression, evaluated using **truth tables**.
    - **Propositions** and **logical operators** ($\wedge$, $\vee$, $\neg$, $\rightarrow$, and $\leftrightarrow$) will influence the sorting order. Objects will be sorted first based on their numerical properties, then by the truth table evaluation of their logical expressions.
    - Encourage innovation in how the truth table and logical expressions are implemented and integrated with the sorting process.
3. **Performance Analysis**:
    - Analyze and compare the time complexity of the implemented sorting algorithms using **Big-O Notation**.
    - Measure performance for different dataset sizes and levels of logical complexity (i.e., simple vs. complex logical expressions).
    - Visualize the performance impact of adding logical expressions and recursion to the sorting process.
4. **Object-Oriented Design**:
    - Use **OOP principles** to design the tool with class objects that best represent your data set and structure.
    - Implement **abstraction**, **inheritance** and **polymorphism** to allow flexibility in adding more sorting algorithms and types of data.
    - Ensure **error handling** for invalid inputs (e.g., malformed logical expressions or improper data formats).
    - A minimum of 4 objects is required.
5. **Data Handling**:
    - Read datasets from **CSV files**, with each row containing objects with numerical and logical fields (e.g., columns for numbers and logical expressions like $p \wedge q$).
    - Use **Pandas** for efficient data manipulation and integration with the sorting and truth table processes.
6. **User Interaction**:
    - Allow users to input datasets manually or through file uploads, select which sorting algorithm to use, and view real-time sorting results and truth table evaluations.
7. **Innovation & Complexity**:
    - Encourage students to develop creative ways to handle complex data objects (e.g., visualize truth table evaluations or recursive processes).
    - Innovation in performance analysis (e.g., visualizing the time taken by recursion) and extensions such as integrating new sorting criteria will be rewarded.

Submission

- Final code to be pushed to Github. Invite your lecturer as a collaborator. The latest commit before the deadline will be considered the submission.
- **Deadline**: The night (**23:59**) before your last lecture.
- Present your project on the last day of classes during class time.

MODUL UNIVERSITY
PRIVATE UNIVERSITY VIENNA

Name: _____

Criteria	Description	Max	Mark
OOP		**15**	
[redacted]			
Data		**14**	
Variable and Data Types	At least 4 of the following data types need to be present within one or more objects with appropriate reasons: Boolean, integer, float, and string, complex, tuple, range, set, dict, type, NoneType, bytes.	4	
Inputs	At least 2 interface inputs with different data types. All button clicks count as 1 input. All string values count as 1 input unless they are cast into their respective data types without error.	5	
Lists, Dictionary, and Tuples	At least 1 appropriate use of lists of **objects** AND (dictionary of objects OR tuple of objects) needs to be implemented.	5	
Algorithms		**55**	
Loop and Recursion	Implementation of at least 1 **loop-based algorithm** and one **recursive algorithm**. Proper use of recursion will be assessed. Both recursion and loop must be implemented for the list of objects in the project.	10	
[redacted]			
Miscellaneous		**16**	
Functions	Appropriate use of functions internally and externally from objects. More than 2 methods within an object before marks can be allocated. (Objects: 3 marks max, External to Objects: 3 marks max)	6	
[redacted]			
Additional		**6**	
Innovation	Creativity in project execution, such as adding new sorting algorithms, visualizing truth table evaluations, or experimenting with innovative data processing features.	3	
Bonus Mark (Advanced Python)	Any show of advanced coding with packages, algorithms, or data structures beyond the syllabus can be awarded bonus marks. Marks are awarded only with proper presentation and sufficient knowledge of the implemented code.	3	
	PROJECT TOTAL	100	

MODUL UNIVERSITY
PRIVATE UNIVERSITY VIENNA

Name: _____

Note: Marks are only awarded if they work. i.e. If the file input and output don't work, you don't get the full marks for that criteria.

Validation criteria	Max	Mark
Group work items	**-50**	
- Missing GitHub repository	-50	
- Missing contribution on GitHub **BEFORE** MOCK exam - Less than 2 proper commits -10 - Less than 5 proper commits -5 - Less than 7 proper commits -3	-20	
- Missing contribution on GitHub **AFTER** MOCK exam (if sufficient contributions were made before the semester break, this section can be ignored) - Less than 3 proper commits -10 - Less than 5 proper commits -5 - Less than 8 proper commits -2		
Run time items	**-50**	
- System crash (-5 per crash)	-50	

	Max	Mark
Project Grand Total = Project total – Validation criteria	100	

Written Component	Max	Mark
Object-Oriented Design	10	
Variable and Data Types	10	
Loop and Recursion	10	
Sorting Algorithm	10	
Searching Algorithm	10	
Truth Table and Logical Complexity	10	
Performance Analysis	10	
Functions	10	
Data Handling	10	
Error Handling	10	
ORAL TOTAL	100	

GP = Group Project Mark
W = Written Mark

	Max	Mark
$Final\ Mark = \frac{\mathrm{GP}\ -(\mathrm{GP}-\mathrm{W})^2}{100}$	100	