Programming for Data Science (24/03/2025)

Upload the solutions to the programming exercises to the following link:
https://evo.di.unipi.it/student/courses/16/exams/VMLXkr0

Exercise 1. (Math, on paper)
Consider the statement: *"It is not the case that all students like logic and mathematics."* and let the domain be the students.

A. Express the statement formally using quantifiers and logical connectives.
B. Apply De Morgan's laws step by step to simplify the formula so that negation is applied only to atomic formulae (i.e., no negation outside quantifiers or compound expressions)

Exercise 2. (Math, on paper)

A password must consist of 4 characters, where each character is either a digit (0–9) or a lowercase letter (a–z) (26 lowercase letters in the English alphabet:).

A. How many different passwords are possible if repetition is allowed, and the password can contain any combination of digits and lowercase letters?
B. How many of these passwords contain only letters?
C. How many of these passwords contain at least one digit?
D. How many different passwords are possible if the password can contain any combination of digits and lowercase letters but repetition is not allowed ?

Exercise 3. (Python 1) Write a Python program that processes a list of numerical data entered by the user. Implement the following functions over the sequence:

1. **Read** a sequence of floating-point numbers from the user, terminated by a sentinel value (e.g., 0). **Store** them in an appropriate **data structure**.
2. Implement a function that calculates the **average** of the numbers.
3. Implement a **recursive** function that calculates the **product** of all elements in the list.
4. Implement an iterative function that finds and returns the **three largest distinct numbers** in the list. If there are fewer than three distinct numbers, return all available unique numbers sorted in descending order.

Exercise 4. (Python 2) Write a Python program that processes textual data from a file. Implement the following functions over the sequence:

1. **Take a file name from** the user and attempt to **open it**. If the file does not exist, handle the exception and prompt the user again.
2. **Read** the content of the file and **count** the total number of **lines, words, and characters**.
3. Implement a function that extracts and prints all **unique words** from the file, **sorted alphabetically**.
4. Implement a function that finds and prints the **most frequently** occurring **word** in the file along with its **count** (i.e., its frequency within the file).