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.

   Let the domain be all students. Define:

   L(x): "x likes logic"
   M(x): "x likes mathematics"

   The statement is: "It is **not the case** that **all** students like logic and mathematics."

   That is: $\neg\forall x(L(x)\wedge M(x))$

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)

   Step 1:

   $\neg\forall x(L(x)\wedge M(x))\equiv\exists x\neg(L(x)\wedge M(x))$

   (using: $\neg\forall x\,\phi(x)\equiv\exists x\,\neg\phi(x)$

   Step 2: (using: $\neg(A\wedge B)\equiv\neg A\vee\neg B$

   $\exists x\neg(L(x)\wedge M(x))\ \equiv\ \exists x(\neg L(x)\vee\neg M(x))$

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 ?

Great question! Let's walk through each part step-by-step.

A. Total number of passwords (with repetition allowed)
Each character in the password can be:

- A digit (0–9): 10 options
- A lowercase letter (a–z): 26 options
  So total options per character: 10+26=3610 + 26 = 36

Since repetition is **allowed** and there are **4 characters**:
Total= 36^4

B. Passwords that contain only letters (with repetition)
Only lowercase letters: 26 options per character.
Letters only=26^4

C. Passwords that contain at least one digit (with repetition)
This is the **complement** of "only letters."
So:
At least one digit=Total passwords−Letters only=36^4-26^4

D. Total number of passwords (no repetition allowed)
Now, repetition is **not allowed**, so each character must be **unique**.
There are still 36 total choices (10 digits + 26 letters), but for each subsequent character, the number of options decreases:
Total=36×35×34×33

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.

```
def read_list():
    result = []
    while True:
        num = float(input("Input a floating point number, or '0' to stop:"))
        if num == 0:
            break
        result.append(num)
    return result

def average(lst):
    sum_list = sum(lst)
    length_list = len(lst)
    average = sum_list/length_list
    return average
```

```
def recursive_product(lst):
    if len(lst) == 1:
        return lst[0]
    else:
        return lst[0] * recursive_product(lst[1:])

def three_largest_distinct_numbers(lst):

    distinct_largest_numbers = []
    for element in lst:
        if element not in distinct_largest_numbers:
            if len(distinct_largest_numbers) < 3:
                distinct_largest_numbers.append(element)
            else:
                for largest_number in distinct_largest_numbers:
                    if element > largest_number:
                        distinct_largest_numbers.remove(largest_number)
                        distinct_largest_numbers.append(element)
                        break
    return distinct_largest_numbers

def three_largest_distinct_numbers_alternative(lst):

    distinct_numbers = set(lst) # remove duplicates
    sorted_distinct_numbers = sorted(distinct_numbers, reverse=True) # sort
    return sorted_distinct_numbers[:3] # take top three from the sorted list

lst = read_list()
print("The average of the numbers: ",average(lst))
print("The product of the numbers: ",recursive_product(lst))
print("Top three largest distinct number:", three_largest_distinct_numbers(lst))
print("Top three largest distinct number:", three_largest_distinct_numbers_alternative(lst))
```

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).

```
def get_file_name():
    while True:
        file_name = input("Please enter the file name: ")
```

```
        try:
            with open(file_name, 'r') as file:
                return file_name
        except FileNotFoundError:
            print("File not found. Please try again.")

def read_file_content(file_name):
    with open(file_name, 'r') as file:
        content = file.read()
    return content

def count_lines_words_characters(content):
    lines = content.splitlines()
    num_lines = len(lines)
    num_words = len(content.split())
    num_characters = len(content)
    return num_lines, num_words, num_characters

def extract_unique_words(content):
    words = set(content.split())
    unique_words = sorted(words)
    return unique_words

def find_most_frequent_word(content):
    words = content.split()
    frequency = {}

    for word in words:
        word = word.lower()  # Normalize to lowercase
        if word in frequency:
            frequency[word] += 1
        else:
            frequency[word] = 1

    most_frequent_word = max(frequency, key=frequency.get)
    return most_frequent_word, frequency[most_frequent_word]

def main():
    file_name = get_file_name()
    content = read_file_content(file_name)

    num_lines, num_words, num_characters = count_lines_words_characters(content)
    print(f"Total lines: {num_lines}")
    print(f"Total words: {num_words}")
    print(f"Total characters: {num_characters}")

    unique_words = extract_unique_words(content)
```

```
    print("Unique words (sorted alphabetically):")
    print(unique_words)

    most_frequent_word, frequency = find_most_frequent_word(content)
    print(f"The most frequently occurring word is '{most_frequent_word}' with a count of {frequency}.")

if __name__ == "__main__":
    main()
```