Programming for Data Science (11/1/2024)

30% of the points are assigned to quality of documentation and/or comments to solutions.
Solutions must include tests of executions of the developed *functions.*

Name files as "<your matricola>_<firstname>_<lastname>_ex1.py" for Exercise 1, and "<your matricola>_<firstname>_<lastname>_ex2.c" for the second exercise.

Upload the TWO files in a folder
(named with your student number and your last name) at the following URL: *Upload here*
(access GDrive using your university credentials)

Exercise 1. (Math, on paper)

Complete the following descriptions for sets of Natural numbers including, respectively, only even numbers and prime numbers:
1a. Even = {x…….. | ……. } // use the congruence relation modulo
1b. Primes= {x …….. | ……..} // use the a divides b (denoted a | b) relation

Formalize in first order logic:
1c. "There is a number that is both even and prime"
1d. "All odd natural numbers are greater than zero"

Let MEn be the set of matrices nxn such that all elements in the matrices are even. Let M1, M2 ∈ MEn
1e. Does M1+M2 ∈ MEn ? Justify your answer
1f. Does M1 * M2 ∈ MEn ? Justify your answer
1g. Is the determinant of M1 even? Justify your answer
1h. Is the rank of M1 at most n-1? Justify your answer

Exercise 2. (Python) Create a Python program that performs basic operations on a list of numbers. Implement the following functions:

- **Sum of Even Numbers**: Write a function to calculate and return the sum of all even numbers in a given list.
- **Count Prime Numbers**: Write a function to count and return the number of prime numbers in a given list.
- **Remove Duplicates**: Write a function to remove duplicate elements from a list and return the modified list.
- **Find Maximum**: Write a function to find and return the maximum value in a list.

Create a list of positive integers, taking input from the user until a -1 is inserted. Handle invalid inputs, such as non-numeric input or an empty list. For prime number checking, define your function for primality testing. Test all the functions on the created list, showing the output of each function.

Exercise 3. (C) Write a C program that dynamically allocates memory to perform various string manipulation operations. Implement the following functions:

1. **Concatenate Strings**: Implement a function to concatenate two input strings dynamically. Use dynamic memory allocation functions (malloc, calloc, or realloc) to allocate memory for the concatenated result.

2. **Reverse String**: Implement a function to reverse a given string dynamically. Allocate memory as needed for the reversed string.
3. **Uppercase Conversion**: Implement a function to convert all characters in a given string to uppercase dynamically. Allocate memory for the resulting uppercase string.
4. **Palindrome Check**: Implement a function to check if a given string is a palindrome (i.e., a sequence that reads the same backward as forward, e.g., madam). Return 1 if it is a palindrome, 0 otherwise.

In the main function, prompt the user to input two strings (string1 and string2). Call each of the implemented functions and display the results.

Memory Cleanup: Free the dynamically allocated memory for each operation before exiting the program.