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

SOLUTIONS

1a. Even = {x ∈ N | x ≡ 0 (mod 2) }
1b. Primes= {x ∈ N | ∀ y (y | x □ y=x or y=1}
1c. Let p(x) denote x is prime, and e(x) denote x is even: ∃ x. (p(X) ∧ e(x)) "
1d. Let o(x) denote x is odd eand let the universe be the natural numbers: ∀ x. (o(x) □ x>0)
1e. Yes, since each element c_{ij} of M1+M2 is the sum of a_{ij} ∈ M1 and b_{ij} ∈ M2 and the sum of even numbers is even
1f. Yes, since each element c_{ij} of M1 * M2 is obtained with sum and product of even numbers
1g. Yes, since it is is obtained with sum and product of even numbers
1h. No, consider the counter-example $[2\ 6\ 4\ 12\ ]$ in general, if you take a non-singular matrix (i.e. a full rank matrix) and multiply all elements by 2, the rank does not change and all elements are now even.

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.

SOLUTION:

```
def sum_even_numbers(l):
    sum = 0
    for elem in l:
        if elem % 2 == 0:
            sum += elem
    return sum

def primality_check(n):
    for i in range(2, n):
        if n % i == 0:
            # not prime
            return False
    # otherwise prime
    return True

def count_prime(l):
    count = 0
    for elem in l:
        if primality_check(elem):
            count += 1
    return count

def remove_duplicates(l):
    new_list = []
    for elem in l:
        if elem not in new_list:
            new_list.append(elem)
    return new_list

def find_max(l):
    max = 0
    for elem in l:
        if elem > max:
            max = elem
    return max

l = [2,4,5,7,9,10,10]
print(f"The sum of the even numbers is: {sum_even_numbers(l)}")
print(f"The prime numbers are: {count_prime(l)}")
```

```python
print(f"The list without duplicates is: {remove_duplicates(l)}")
print(f"The max number in the list is: {find_max(l)}")
```

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.

SOLUZIONE:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

char* concatenateStrings(const char* str1, const char* str2) {
 int length1 = strlen(str1);
 int length2 = strlen(str2);

 char* concatenated = malloc((length1 + length2 + 1) * sizeof(char));
 if (concatenated == NULL) {
  printf("Memory allocation failed\n");
  exit(EXIT_FAILURE);
 }

 // copy str1 into concatened
 for (int i=0; i<length1; i++)
  concatenated[i] = str1[i];

 // copy str2 into concatened, at the end of str1
 for (int i=0; i<length2; i++)
  concatenated[length1+i] = str2[i];

 // add the termination character at the end of concatened
```

```
 concatenated[length1+length2] = '\0';

 return concatenated;
}


char* uppercaseString(const char* input) {
 int length = strlen(input);

 char* uppercase = malloc((length + 1) * sizeof(char));
 if (uppercase == NULL) {
  printf("Memory allocation failed\n");
  exit(EXIT_FAILURE);
 }

 for (int i=0; i<length; i++) {
  uppercase[i] = toupper(input[i]);
 }

 uppercase[length] = '\0';

 return uppercase;
}


char* reverseString(const char* input) {
 int length = strlen(input);
 char* reversed = malloc((length + 1) * sizeof(char));
 if (reversed == NULL) {
  printf("Memory allocation failed\n");
  exit(EXIT_FAILURE);
 }

 for (int i=0; i<length; i++) {
  reversed[i] = input[length - 1 - i];
 }
 reversed[length] = '\0';

 return reversed;
}

int isPalindrome(const char *str) {
 int length = strlen(str);
 int i, j;

 for (i = 0, j = length - 1; i < j; i++, j--) {
  if (str[i] != str[j]) {
   // Not a palindrome string
   return 0;
```

```
  }
 }

 // Is palindrome
 return 1;
}

int main() {
 char string1[32];
 char string2[32];
 printf("type here 2 strings:\n");
 scanf("%s",string1);
 scanf("%s",string2);

 printf("String 1: %s\n", string1);
 printf("String 2: %s\n", string2);
 printf("Concatenated: %s\n", concatenateStrings(&string1, &string2));
 printf("First Reversed: %s\n", reverseString(&string1));
 printf("First Uppercase: %s\n", uppercaseString(&string1));

 return 0;
}
```