Algorithms and Programming
Laboratory number 02

Exercise 01

Re-write (and clean-up) the C program of
Laboratory Number 01 Exercise 04
dealing with matrix multiplication, using the following functions:

1.
   ```
   void readDim(int *, int *);
   ```
   read the number of row and the number of column of a matrix

2.
   ```
   int checkDim(int, int, int, int);
   ```
   it receives the size of the two matrices and check for their compatibility

3.
   ```
   void readMatrix(float [][MAX], int, int);
   ```
   read the matrix m of size r x c from standard input

4.
   ```
   void computeProduct(float [][MAX], int, int,
                       float [][MAX], int, int,
                       float [][MAX], int *, int *);
   ```
   compute the product
   m3 = m1 x m2.

5.
   ```
   void writeMatrix(float [][MAX], int, int,
                    float [][MAX], int, int,
                    float [][MAX], int, int);
   ```
   print-out the matrix m1, m2 and the result m3 on standard output.

Exercise 02

A file contains a text with an unknown number of rows, but with lines of maximum length equal to 100 characters.

Write a C program able to find and count-up the number of:
- horizontal sequences made up of 5 adjacent 'h' characters ('h' like in "horizontal") on the same row
- vertical sequences made up of 5 adjacent 'v' characters ('v' like in "vertical") on the same column.

Notice that, as the length of the file is unknown, it cannot be stored entirely at the same time into the main memory (i.e., it can only be stored into the main memory a piece at a time).
Moreover, it is not possible to read the file more than once.

The program:
- receives on the command line the name of the file
- prints out (on standard output) the number of times the "hhhhh" sequence appears on a row, and the number of times the sequence "vvvvv" appears on a column.

Example

Let the input file be the following one:

```
Regression file:
Line with two hhhhhorizontal sequences hhhhh!
From hereinafter  vv
v all rows        vv
```

```
v include         vv
v two vertical    vv
v sequences       v
v and one horizontal ---> hhhhh!
```

The output will be:

```
Horizontal Sequence: 3
Vertical Sequence  : 2
```

Suggestions

- Read the file on a line-by-line basis using fgets, like
  while (fgets (...) != NULL) { ... }
- Looking for horizontal sequences on "h"s is straightforward just
  storing single line of the file into the main memory.
  To check for vertical sequences of "v"s there are two main
  possibilities:
  a. Work on single lines, keeping an array of (100) counters
  b. Work on a matrix which stores 5 rows of the file at the same
     time. Once the check has been performed scroll ("shift") all
     rows up and read the last one from file.

Exercise 03

A C program is run with 4 parameters on the command line:

```
string1 file1 string2 file2
```

where file1 and file2 are file names and string1 and string2 are C
strings.
It makes a copy of the content of file1 (input file) into file2
(output file) by substituting each sequence of characters equal to
string1 with the string2 sequence.

Notice that:
* file1 contains an unknown number of rows, but each row is at
  most 100 characters long
* string1, string2 and all C strings in file1 are at most 20
  20 characters long
* string1 and string2 are proper C strings, i.e., they to not
  contain spaces, tabs, etc.
* all rows that have to be stored into file2 are also shorter than 100
  characters.

Example

Let us suppose to run the program with the following parameters:

```
test file1.txt check file2.txt
```

and that file1.txt is the following:

```
Regression file.
This file is to test the program.
In the previous line we should have a substitution (testWITHcheck).
Here we should not have it: t-e-s-t.
```

The following file2.txt has to be generated:

```
Regression file.
This file is to check the program.
In the previous line we should have a substitution (checkWITHcheck).
Here we should not have it: t-e-s-t.
```

Suggestions

- Read the file on a line-by-line basis using fgets, like
  `while (fgets ( ... ) != NULL) { ... }`
- Search string1 as a sub-string in each row
- For each hit, before moving on with the next search, write into file2 everything coming before string1 and then string2 instead of string1.