PROGRAMMING TECHNIQUES, A.A. 2023/2024

Laboratory 5

Objectives

- Solve **iterative** numerical and text processing problems, using **arrays** (mono-dimensional arrays and matrixes) (*C3b- Problem solving with arrays: part II and III*)

Technical content

- I/O basics
- Functions
- Conditional and **iterative** problems
- Operations with **arrays** (of int, float and char)

Exercise 1.

Category: problems with numerical matrixes, verification, and selection problems.

Championship

In a championship `n` (max 20) teams play for `m` (max 20) days. The outcome of the matches is stored a matrix of `n` × `m` integers, each of which can only be either 0, 1 or 3. Each `i-th` **row** of the matrix represents the points acquired by the i-th team in the **m** days of the championship. Each `j-th` **column** of the matrix represents the points acquired by the **n** teams in the matches of the `j-th` day. The points acquired are 3 for a won game, 1 for tied and 0 for a defeat, respectively.

Write a program C that:

- acquires the content of the matrix from a text file whose name (20 characters maximum) is read from the keyboard, defining a suitable format for the file. For example, you may assume that the first line reports the values `n` and `m`, separated by a space, and each following line reports a row of the matrix, with spaces as separators:

```
6 4
0 1 1 0
3 1 0 1
1 0 1 1
1 3 1 0
1 0 1 3
1 3 3 3
```

- for each **matchday** of the championship, print the index (i.e., *the corresponding row number*) of the team that is currently leading the championship (i.e., the team that has the highest sum of points since the beginning of the championship). In case there is more than one leading team, the program should print the first that is found.

Exercise 2.
Category: text processing problems and text encoding.

Re-encoding of a text based on a conversion table.

A file (`source.txt`) contains a text made up of an indefinite number of lines, each with a maximum length of **200** characters. A second file (`dictionary.txt`) is organized as follows:

- in the first line there is a positive integer number `S` (≤ 30), which indicates the number of possible re-encodings (i.e., string replacements) that are contained in the dictionary
- in the following `S` lines of the file, one per line, there is a `<replaced> <original>` pair, representing a possible re-encoding: more specifically, `<original>` is the original sequence of characters and `<replaced>` the corresponding replacement, in a `$<integer>$` format.

The purpose of the program is to re-encode the first text file (`source.txt`), by replacing the original sequences of characters with the corresponding replacements, based on the conversion table specified by `dictionary.txt`. In case there is more than one replacement possible for a given sequence of characters, the program should apply the first replacement that is found in the dictionary file. The result of the re-encoding should be saved on a third file, `output.txt`.

Example
Content of the file `source.txt`:

```
apelle figlio di apollo
fece una palla di pelle di pollo
tutti i pesci vennero a galla
per vedere la palla di pelle di pollo
fatta da apelle figlio di apollo
```

Content of the file `dictionary.txt`:

```
9
$11$ pelle
$2$ pollo
$333$ palla
$41$ alla
$5078$ tta
$6$ tti
$7$ ll
$81$ er
$900$ ere
```

The file `output.txt` should be:

```
a$11$ figlio di a$2$
fece una $333$ di $11$ di $2$
tu$6$ i pesci venn$81$o a g$41$
p$81$ ved$81$e la $333$ di $11$ di $2$
fa$5078$ da a$11$ figlio di a$2$
```

Exercise 3.

Category: problems with numerical matrixes and text processing.

Write a C program that allows the user to perform sequential rotation operations of **P** positions on specified **rows** and / or **columns** of an input integer matrix. The rotations are to be interpreted as *circular* both on the rows and on the columns (see definition given in Lab. 4 ex. 2). Consider the **rows** and **columns** numbered starting from 1 (NB: in C, indexes of a matrix start from 0!).

The program should:

- read the initial matrix (max **30** x **30**) from a file, whose name (20 characters maximum) is acquired from the keyboard. The file should contain 2 integer values "`nr nc`" on the first line, indicating the number of rows "`nr`" and columns "`nc`" of the matrix. The following `nr` lines of the file, one per row of the matrix, should report `nc` integers each, separated by spaces.
- repeatedly acquire a command (string of maximum 100 characters, containing spaces) from the keyboard, in the format
  `<selector> <index> <direction> <locations>`
  Where `<selector>` is either the string "**row**", "**column**" or "**end**", respectively indicating whether the user wants to perform the rotation on a row ("row"), on a column ("column"), or terminate the program ("end").
  `<index>` is the index of the selected row/column, starting from 1
  `<direction>` is either the string "right", "left", "up" or "down", indicating the direction of the rotation
  `<locations>` is a positive integer indicating the number of positions `P` of the rotation.
  The resulting matrix after each sequential rotation should be printed on the screen.

  NB. *To perform the rotation, implement a function that generalizes what you have already implemented in Lab 4, ex.2.*

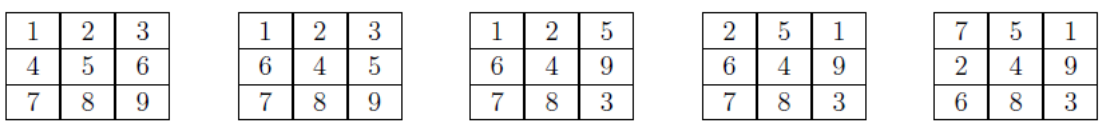
Example of input file:

```
3 3
1 2 3
4 5 6
7 8 9
```

Example of sequential commands acquired from keyboard:

```
row 2 right 1
column 3 down 2
row 1 left 4
column 1 up 2
end
```

The results of the rotation operations are represented by the following figure:



original matrix | row 2 right 1 | column 3 down 2 | row 1 left 4 | column 1 up 2