Algorithms and Programming
Laboratory number 09

Exercise 01

A file includes a labyrinth with the following specifications:
- the first line of the file specifies the number of row R and the number of columns C of a matrix
- the following R lines specify the matrix rows (each one with C characters), where each
  - '@' indicates the initial position of a human being
  - ' ' represents corridors (empty cells)
  - '*' represents walls (full cells)
  - '#' represents exit points.

Supposed only one person is present in the maze in the initial configuration.

The following is a correct example of such a maze:

```
12 10
**********
*          *
*    **@* *
* * ** * *
* ****** *
*    **    *
* *      ***
* ** * ***
* *   * * *
* * ** * *
*    **    *
******#***
```

Write three recursive functions to find:
- one escape path
- all escape paths
- the shortest escape path

from the maze.
Print out the solution (or all solutions) on standard output.

Observation

To avoid extremely long running times, check the program on small mazes or on mazes on which the number (and length) of escaping paths is limited.

Exercise 02

The directory
lab09Library
includes two stack libraries:
- The first one (impl1) implemented on a dynamic array.
- The second one (impl2) implemented on a dynamic list.

Both of them include the following files:
- client.c
  the client using the stack
- stack.c, stackPublic.h, stackPrivate.h
  implementing all main stack functions (push, pop, etc.)
- item-int.c, item-string.c, item-struct.c
  and corresponding *.h files
  implementing the data type (i.e., enabling a stack on integer, string and structure items)
- util.c, util.h
  implementing a few library functions (such as malloc, free, etc.).

Perform the following operations:
- Select one of the two packages (the implementation on the dynamic array or on the dynamic list)
- Create a project in the Code-Block environment, "enabling" the stack with C structure elements
- Build the application, run to run it, understand it, and check if it runs correctly
- Add one more feature into the menu, which displays the content of the stack structure in FIFO order.
  Use a recursive function to visit either the array or the list.

Elective (optional)

Add the same feature to the other version of the application (i.e., the one implemented on a list or array).