C++ 2021B - MTA - Exercises

Requirements and Guidelines

The exercises in the course would require you to implement a tetris game as a console application.

Note: the exercise should be implemented in Visual Studio 2019 or later (e.g. VS 2021), with standard C++ libraries and run on Windows with Console screen of standard size (80*25), using gotoxy for printing at specific location on screen (X, Y), using _kbhit and _getch for getting input from the user without blocking, and Sleep for controlling the pace of the game. Submission is in MAMA, as a single zip file containing only the code and the vcproj and sln files + readme.txt with IDs -- but without the DEBUG folder and without any compiled artifact.

Exercise 1

In this exercise you will implement the basic tetris game for two human players.
Read the tetris rules in the wiki link above.
You should support all shape combinations of four squares.
There should be two boards, each of size: 18 squares height * 12 squares width.
Keys:

	Left Player	Right Player
LEFT	a or A	j or J
RIGHT	d or D	l (small L) or L
ROTATE clockwise	s or S	k or K
ROTATE counterclockwise	w or W	i or I (uppercase i)
DROP	x or X	m or M

Menu

The game shall have the following entry menu:
(1) Start a new game
(2) Continue a paused game
(8) Present instructions and keys
(9) EXIT

⇒ NOTE: option (2) would be presented only if the last game played is paused! ⇐

Pausing a game

Pressing the ESC key during a game pauses the game, clears the screen* and presents the main menu with option (2) being presented!
* for clearing the screen you are allowed to use: `system("cls");`

When a player loses there should be a message announcing the winner and then after pressing *any key* the main menu shall be presented.

Bonus points: colors, nice working game, managing score (your choice how to manage that).

Notes on Ex1

1. If you decided to add colors (as a bonus feature) please add an option in the menu to run your game with or without colors (the default can be to use colors, but the menu shall allow a switch between Colors / No Colors) - to allow proper check of your exercise in case your color selection would not be convenient for our eyes. The game MUST work properly in the No Colors selection.
2. Scores are a bonus feature, if you decide to add it the decision regarding score formula is yours. Note that winning the game is not associated with the points: when a player reaches the max height with blocks this player loses and the other one wins. The only case where points can be relevant for winning decision is when both players lose at the exact same time, if you are not adding points this would be a tie, if you add points, the player with higher score would win and it may be a tie only if both lose and there is also tie in the score.
3. Please indicate inside your readme.txt file the bonus additions that you implemented.

Exercise 2

In this exercise you will implement the following additions to your game:

A BOMB

A bomb is a single square block (select your char representation).
When a bomb reaches its position on top of any other square, it explodes and all squares that are in a block distance of 4 squares disappear (4 squares to each direction).
The chances for a bomb to appear are 5% per each new shape that should appear.

Game against the Computer

Game shall allow now to play against the computer. The computer shall have 3 levels:
(a) BEST (b) GOOD and (c) NOVICE
BEST - should calculate the exact position for each move
GOOD - should miss occasionally (randomly, once in ~40 moves)
NOVICE - should miss occasionally (randomly, once in ~10 moves)
=> The BEST level would also use the Bomb wisely, the other levels would not

There shall be three options to play:
2 human players
Human vs. Computer - Human will always be on the left
2 Computer players

Menu

The game shall have now the following entry menu:
(1) Start a new game - Human vs Human
(2) Start a new game - Human vs Computer
(3) Start a new game - Computer vs Computer
(4) Continue a paused game
(8) Present instructions and keys
(9) EXIT

Note

You should change your original code, to use new materials that we learned - where appropriate.

Exercise 3

In this exercise you will implement an option to run a game from files and to record a game into files, mainly for testing. This would work as following:

- There would be a file with list of blocks per player:
  a.blocks *and* **b.blocks**
  each file would contain a list of blocks and the exact position they should appear on the top of the screen
- There would be a file with list of moves per player:
  a.moves *and* **b.moves**
  each file would contain a list of moves with the exact "cycle" when the move is taken
- There would be a file with the expected result of the game, called **expected.result**, the file shall include the following information:
  the winner (A == left, B == right, or TIE)
  the "cycle" number from beginning of the game when there was a winner,
  number of lines in the winner's board (highest point in board, if a tie - it will be the height of the board)

You have the freedom to decide on the exact format of the files but without changing the above info or adding unnecessary additional info.

You should provide at least 3 examples, under folders with the exact names: **game1**, **game2**, **game3** + instructions in a dedicated readme file called **file_format.txt** explaining your files format, to allow creation of new files or update of existing files.

Running this options would be done from the command line with the following parameters:

```
tetris.exe [-path <folder>] -load|-save [-silent]
```

If the -path <folder> is missing then the current working directory would be used.

The *-load/-save* option is for deciding whether the run is for saving files while playing or for loading files and playing the game from the files.
In the -load mode there is NO menu, just run the loaded game as is **and finish!** Also you should ignore any user input, including ESC - there is no support for ESC in load mode!
In the -save mode there is a menu and the game is the same as in Ex2, except that files are saved. Note that each new game overrides the files of the previous game, i.e. we always keep the last game saved (you can always take the files and copy them to another folder).

The *-silent* option is relevant only for load, if it is provided in save mode you should ignore it, if it is provided in load mode, then the game shall run without printing to screen and just testing that the actual result corresponds to the expected result, with a proper output to screen (test failed / passed). In silent mode you should better avoid any unnecessary sleep, it should run as fast as possible.

Note:
You should still support running your game in "simple" mode, without any command line parameters, as in Ex2:
`tetris.exe`
In which case it will **not save or load files** and would behave as in Ex2.

Exercise 4

Exercise 4 is a separate exercise built as a rehearsal for the exam.
It would be published separately.