Vanier College

This is an individual assignment. You are permitted to get help ONLY from the teacher.

Objectives
In this assignment, you will use and practice the principles of functions (void functions and functions that return a value), call-by-value, call-by-reference and multidimensional arrays as well as input and output streams.

Problem statement
This assignment is about the implementation of the game “Funny Boxes”. You are asked to implement a turn-based, two-player game as explained in the class.

Rules

- The following letters denote each player: **H** (for human, and **P** (for computer).
- Player H always plays first.
- When a player takes a turn, the coordinates (i and j) of the desired grid box should be entered. Using *cin* for player H, and ***randomly*** generated for player **P.**
- You should prompt the user to enter i and j (e.g display “player H>” before *cin*)
- When player P takes a turn, your program should display the selected coordinates. E.g player P selected box 2 3
- You should write in the selected box the letter that reflects the current player.
- Before each turn, you should display the current state of the game board.
- A player should select only a free box (a box containing a dash). Hence, you must verify the content of each selected box (for both players) before overriding its value.
- You must validate the coordinates entered by player H.
- See the possible cases below to declare a winner,
- **When the game is over, you must write the final board to a file.**

Game Board
The game is a **5**x**5** grid. Characters “|” and “_” are used to draw the desired grid. At first, each case should be initialized by the dash character “-“. The rows and columns of the grid are labeled with numbers 0 to 4.

Functions to be implemented

You are free to declare additional functions. However, you MUST write the following functions:

```
initializeBoard
      Description: initialize the game board by putting dashes in all
                   the boxes.
      Input: the game board.
      Output: the game board (call-by-reference).

displayBoaord
      Description: displays the board.
      Input: the game board.

isInputValid
      Description: checks if the entered coordinates are valid (i and
      j).
      Input: the coordinates.
      Output: boolean.

isBoxFree
      Description: checks if a box is free or not.
      Input: the value of the selected box.
      Output: boolean.

isGameOver
      Description: determines whether if all the boxes have been closed
                   or not. This function determines the winner as well.
      Input: the game board.
      Output: boolean.
```

Assignment submission requirements and procedure

- Before submitting, your code MUST compile and execute, do not submit a code that does not compile.
- You have to submit your assignment before the deadline. Late assignments are not accepted.
- The file must be a **.zip** file containing your Visual Studio project and your code. Do not submit ONLY your .cpp file.
- Before zipping your files, please REMOVE the following: **.sdf** and **ipch** folder.

Initial state of the game board:

```
   0 1 2 3 4
  -----------
0 |-|-|-|-|-|
  -----------
1 |-|-|-|-|-|
  -----------
2 |-|-|-|-|-|
  -----------
3 |-|-|-|-|-|
  -----------
4 |-|-|-|-|-|
  -----------
```

Typical output:

```
Player H> 1 2
   0 1 2 3 4
  -----------
0 |-|-|-|-|-|
  -----------
1 |-|H|-|-|-|
  -----------
2 |-|-|-|-|-|
  -----------
3 |-|-|-|-|-|
  -----------
4 |-|-|-|-|-|
  -----------

Player P> 3 4
   0 1 2 3 4
  -----------
0 |-|-|-|-|-|
  -----------
1 |-|H|-|-|-|
  -----------
2 |-|-|-|-|-|
  -----------
3 |-|-|-|-|P|
  -----------
4 |-|-|-|-|-|
  -----------
```

Possible states to win the game
X = {H, P}, X can be P or H.

Case 1:
```
    0 1 2 3 4
   -----------
0  |X|-|-|-|-|
   -----------
1  |-|X|-|-|-|
   -----------
2  |-|-|X|-|-|
   -----------
3  |-|-|-|X|-|
   -----------
4  |-|-|-|-|X|
   -----------
```

Case 2:
```
    0 1 2 3 4
   -----------
0  |-|-|-|-|X|
   -----------
1  |-|-|-|X|-|
   -----------
2  |-|-|X|-|-|
   -----------
3  |-|X|-|-|-|
   -----------
4  |X|-|-|-|-|
   -----------
```

Case 3:
In this case, 8 adjacent boxes should be closed by a given player, for example:
```
    0 1 2 3 4
   -----------
0  |-|-|-|-|-|
   -----------
1  |-|X|X|X|X|
   -----------
2  |-|X|X|X|X|
   -----------
3  |-|-|-|-|-|
   -----------
4  |-|-|-|-|-|
   -----------
```