Tic Tac Toe

Implement a Tic Tac Toe game with a command line interface. Players will be able to take turns and undo past turns.

Implementation main.cpp

Contains the main game loop. Players will be able to enter their move in a format "move x y", where x and y are the coordinates of the target cell. Also, players will be able to undo the latest move (of their opponent) by entering "undo". The player board will be displayed at the beginning of the game and after every valid move. After every valid move the win condition has to be checked for the last player who made the move (this function is provided to you in GameState.cpp file). If the win condition is reached a winner will be declared and the program will be terminated. The same goes for if the game ends in a draw.

Hint: Implement main last.

MoveStack.h

A regular stack that stores instances of the Move structure. All past moves are stored here. The stack is required in order to be able to undo moves. In the template, you're provided with public functions. Feel free to add any private variables/functions, but leave public function prototypes unchanged. Unlike the previous labs, you will need to implement every function inside the MoveStack.h file (you can inline the definitions or add them after the class declaration).

- MoveStack() Initializes the stack.
- ~MoveStack() Cleans up the dynamically allocated memory (if any). Must be defined, even if empty.
- int getSize() Returns the size of the stack.
- Move top() Returns a move on top of the stack (but does not remove it).
- void push(Move move) Adds the move to the top of the stack.
- void pop() Remove a move from the top of the stack (but does not return it).

GameState

The GameState class represents the current state of the game. It stores the current board in the "boardState" variable and past moves in the "moveStack" variable.

- GameState() Initializes the board.
- int getCurrentPlayer() Return the player that has to make the next turn. Hint: use the size of the stack to determine who's move it is. Extra hint: odd/even.

- int addMove(Move move) Updates the state of the board and stack of the past moves given a move of the current player. If the turn is invalid (the target position is not empty) it returns -1. If all positions are filled, it returns 0. If the move was successfully done and there are moves available, it returns 1. Hint: use the size of the stack to determine if all positions are filled.
- bool undoLast() Undoes the last turn by changing the board state to the previous one and removing the last move from the stack. Returns true if the move was removed, false if there are no moves to undo.
- void displayBoardState(std::ostream& out) Prints the board state to the "out" stream.

Sample execution (note that _ is an underscore, not a dash).

```
___
___
___
Player 0 make a turn.
move 1 1
___
_x_
___
Player 1 make a turn.
move 1 2
___
_xo
___
Player 0 make a turn.
undo ___
_x_
___
Player 1 make a turn.
undo ___
___
___
Player 0 make a turn.
undo
No moves to undo.
Player 0 make a turn.
move 1 1
___
_x_
___ Player 1 make a
turn.
move 1 2 ___
_xo
```

```
___ Player 0 make a
turn.
move 2 2 ___
_xo
__x Player 1 make a
turn.
move 1 2
Incorrect move. Please try again.
Player 1 make a turn.
move 2 2
Incorrect move. Please try again.
Player 1 make a turn.
move 0 2
__o
_xo
__x Player 0 make a
turn.
undo ___
_xo
__x Player 1 make a
turn.
move 0 0
o__ _xo
__x Player 0 make a
turn.
move 2 0 o__ _xo x_x
Player 1 make a turn.
move 0 2 o_o _xo x_x
Player 0 make a turn.
move 2 1
o_o _xo
xxx

Player 0 won!
```