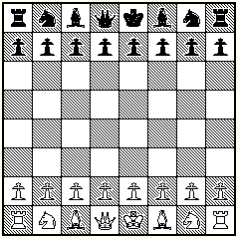
Full Tilt Chess

Pretend that you are working at Mapleton Hill Media, and taking over a half-finished project from your former colleague (sadly, she was hit by a bus). You are finishing software to allow two individuals to play competitive chess for money for a project of dubious legality for some enterprising waste management consultants from New Jersey.

All simulations take place on a chess board (class name: ChessBoard) that is a grid consisting of length X, and height Y – both of which are integers. Chess pieces can be placed on the board at a given (x,y) coordinate pair with (0, 0) being in the lower left-hand corner of the board as seen in the following illustration:

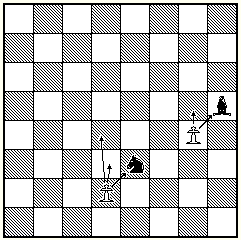
(7, 7)



(0, 0)

Pieces are either Black or White. Black pieces typically start at row x=7 and x=6, whereas white pieces typically start at rows x=0 and x=1. That said, you can set up a board with many initial configurations to replay famous chess games (that last bit might be a paradox).

Additionally, Pieces can be given two commands: move and capture (we will ignore capture for this exercise). Each piece has unique movements, but we are going to focus on commands for pawns. For our limited implementation, Pawns can only more forward one space (toward their opponents side of the board) and can only capture in a forward and diagonal direction as seen in the next illustration.



Your task is to get all unit tests passing for both the chess board and the pawns in the file “PawnSpecs.cs” (all these test have been tested with our working code and should pass once you have coded it correctly).

Good luck, and please reach out to us if you have any questions!