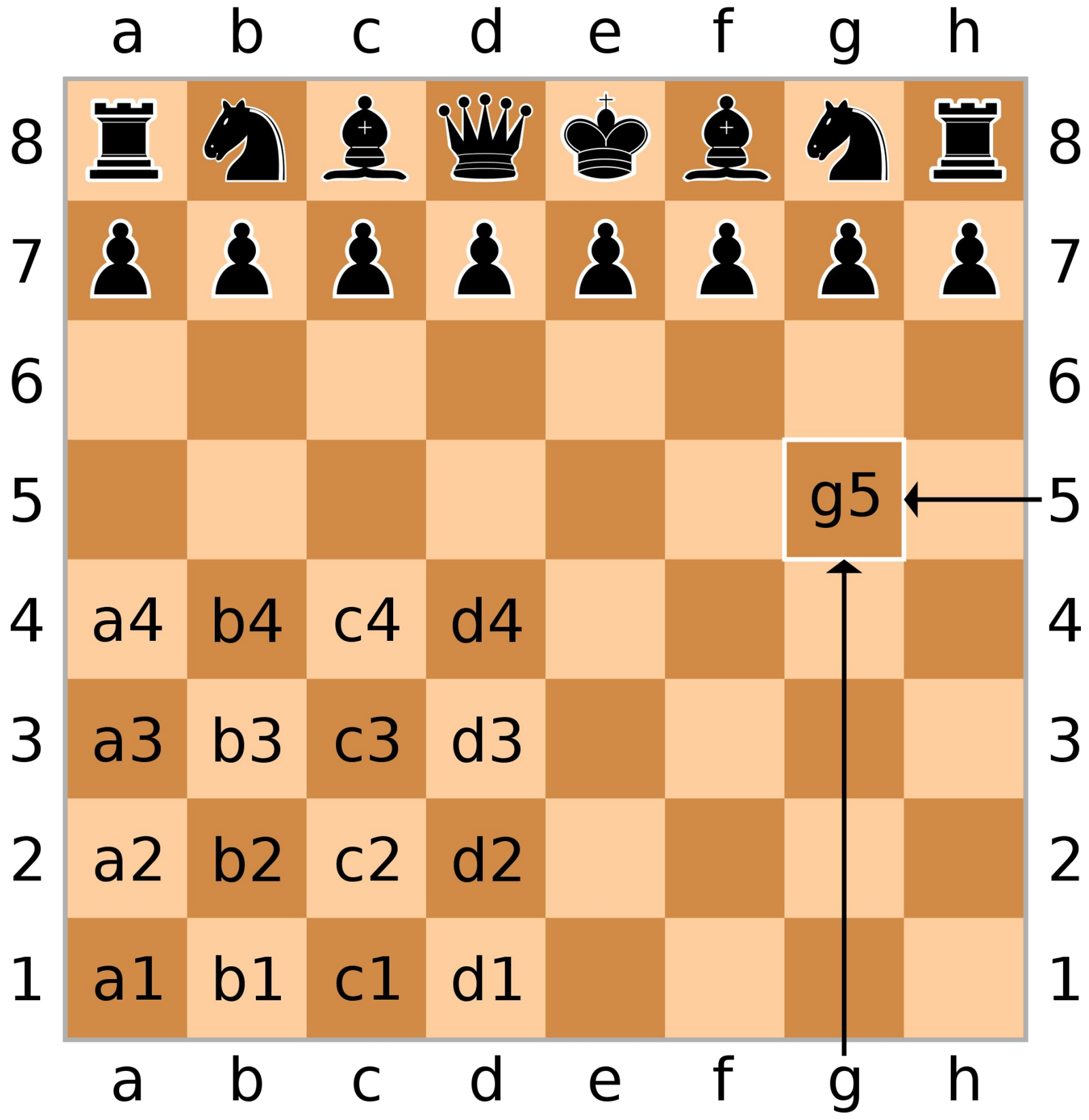
Chess Validator

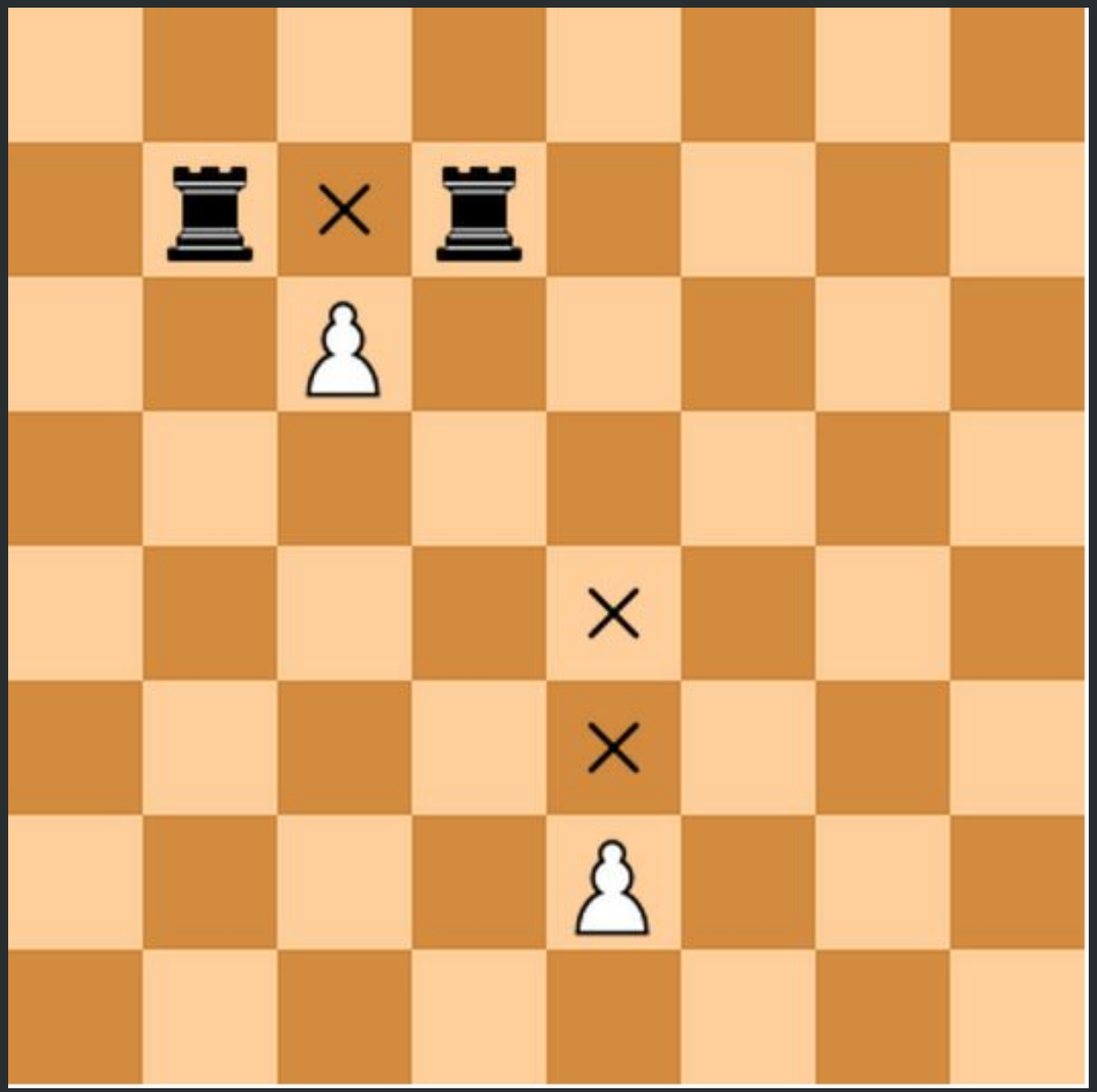
#Let's play

We are building a chess validator

First let's talk about chess.

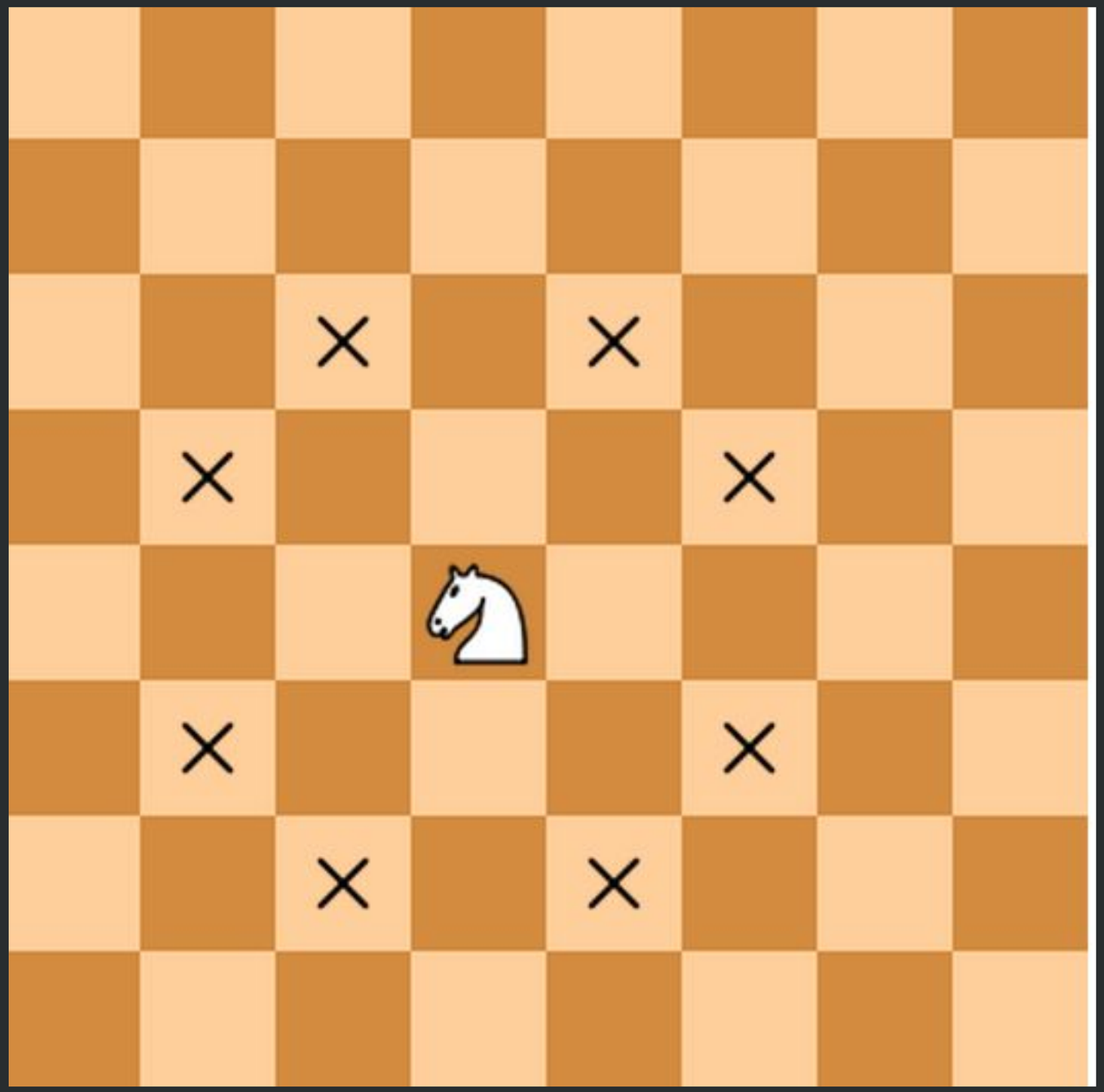

a b c d e f g h
8 7 6 5 4 3 2 1
g5
a4 b4 c4 d4
a3 b3 c3 d3
a2 b2 c2 d2
a1 b1 c1 d1

A pawn moves straight forward one square, if that square is vacant. If it has not yet moved, a pawn also has the option of moving two squares straight forward, provided both squares are vacant.

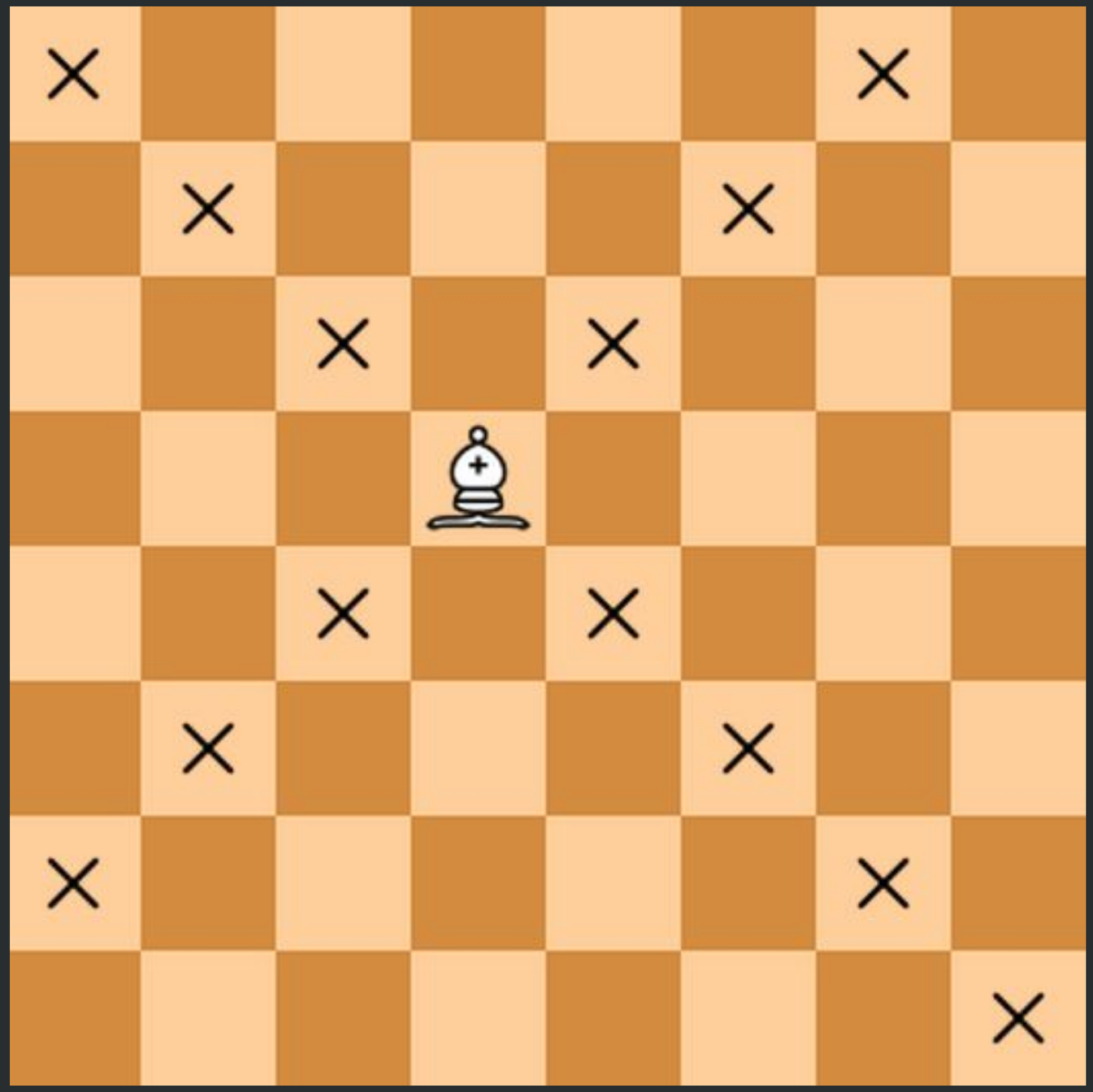


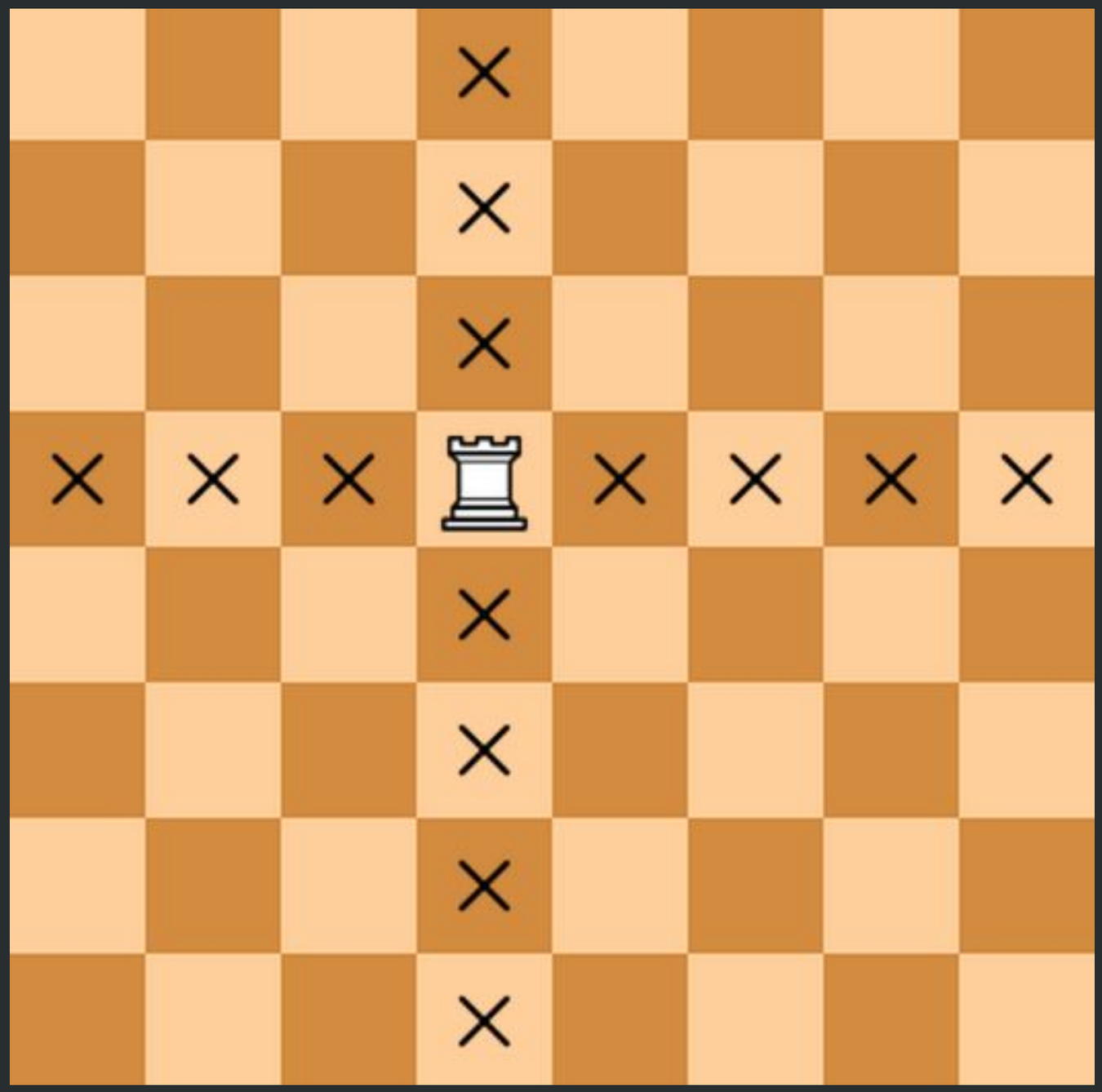
The pawn at the top can also take either black rook.

The knight is not blocked by other pieces: it jumps to the new location.

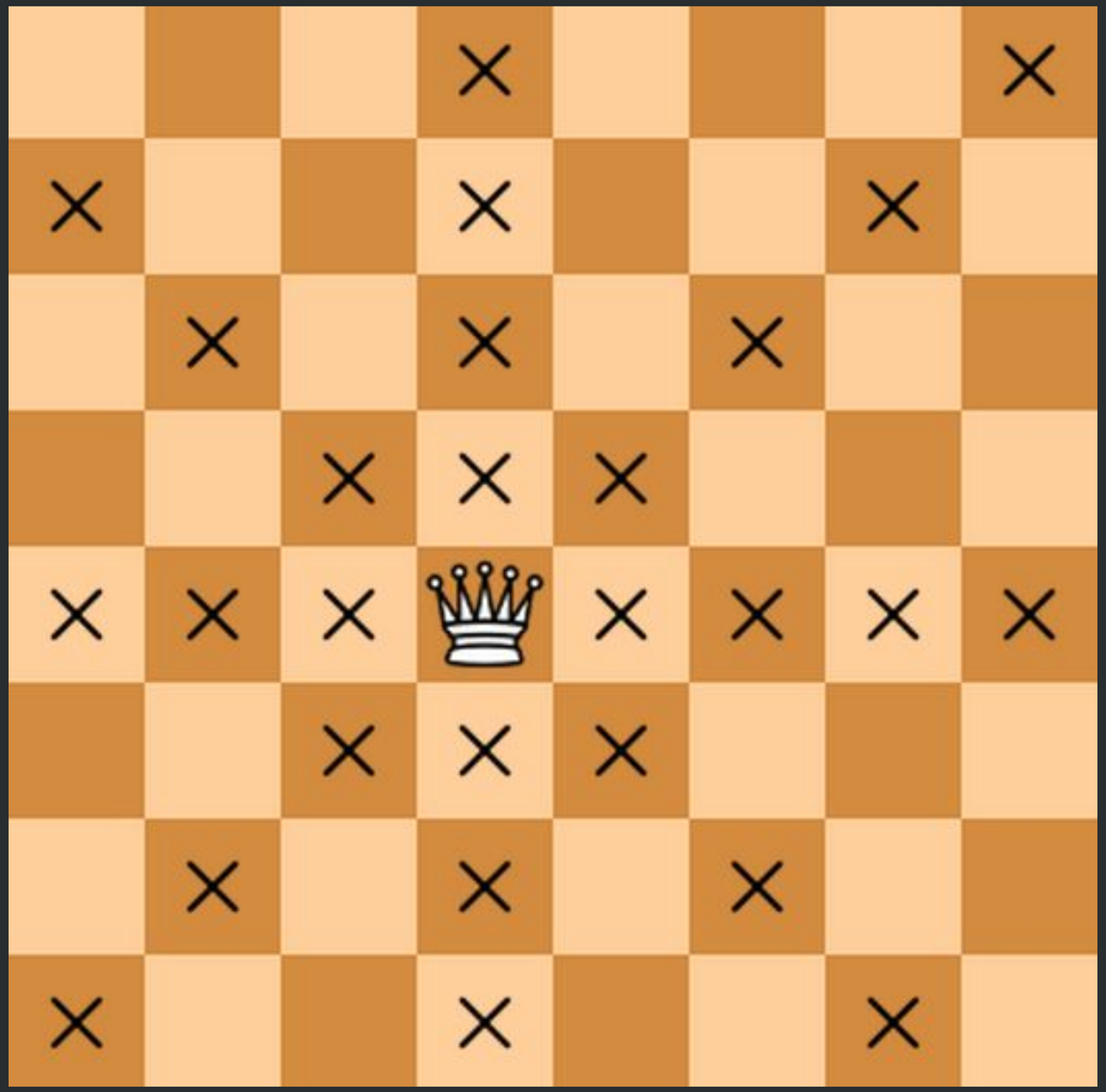


A bishop moves any number of vacant squares in any diagonal direction.

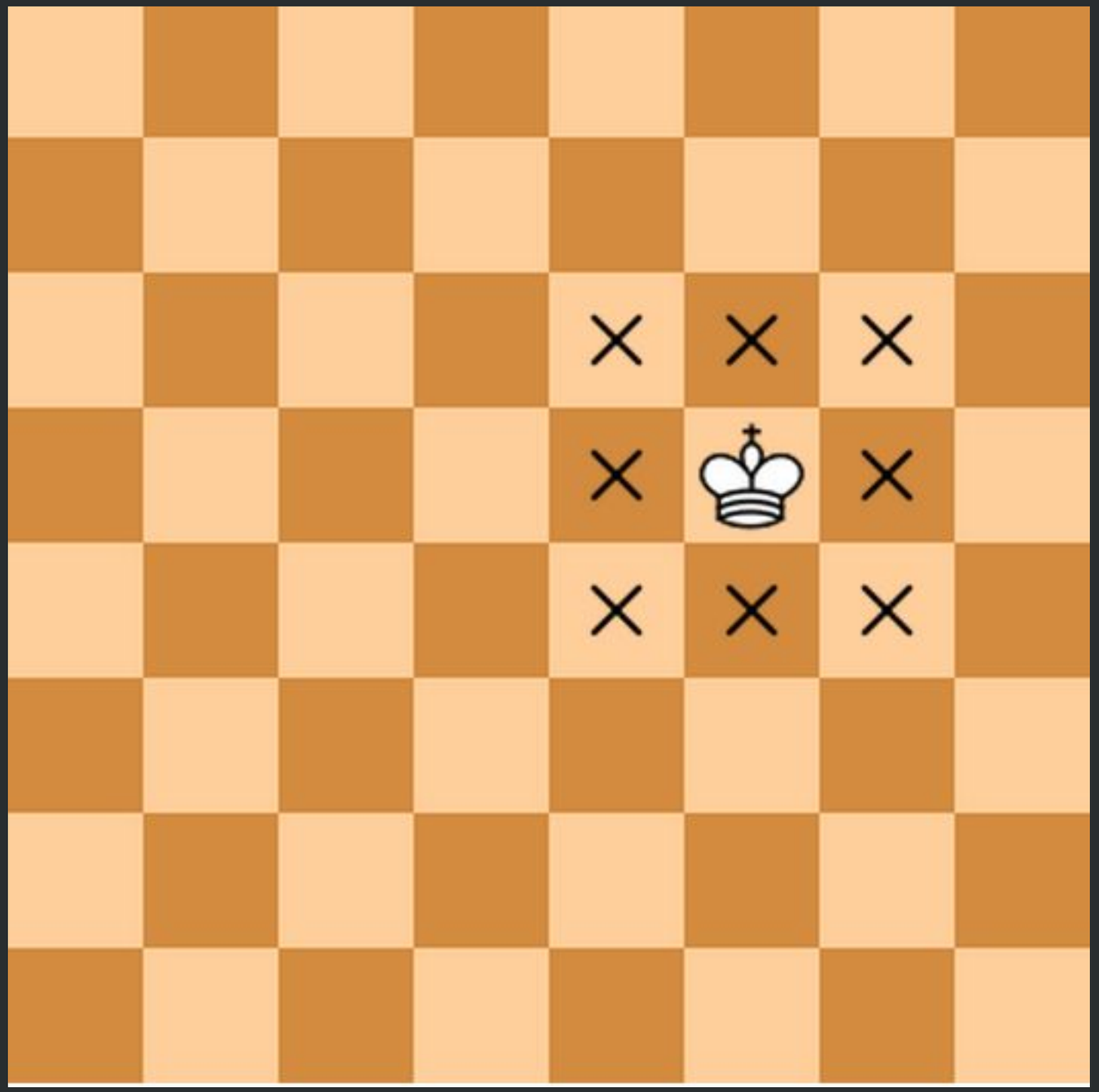




A rook moves any number of vacant squares in a horizontal or vertical direction.



The queen moves any number of vacant squares in a horizontal, vertical, or diagonal direction.



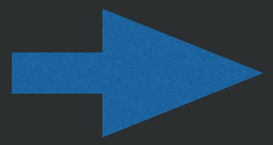
The king moves exactly one square horizontally, vertically, or diagonally.

Read a file with a list of moves.

```
a2 a3
a2 a4
a2 a5
a7 a6
a7 a5
a7 a4
a7 b6
b8 a6
b8 c6
b8 d7
e2 e3
e3 e2
```

Tell the user whether the moves are valid or not.

a2 a3	LEGAL
a2 a4	LEGAL
a2 a5	ILLEGAL
a7 a6	LEGAL
a7 a5	LEGAL
a7 a4	ILLEGAL
a7 b6	ILLEGAL
b8 a6	LEGAL
b8 c6	LEGAL
b8 d7	ILLEGAL
e2 e3	LEGAL
e3 e2	ILLEGAL



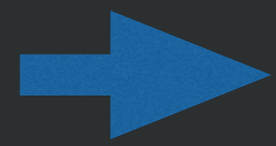
Only consider the pieces' starting position (every move is the piece's first move)

Start with Rooks.

Chess validator

```
bR bN bB bQ bK bB bN bR
bP bP bP bP bP bP bP bP

-- -- -- -- -- -- -- --

-- -- -- -- -- -- -- --

-- -- -- -- -- -- -- --

wP wP wP wP wP wP wP wP
wR wN wB wQ wK wB wN wR
```

```
a2 a3
a2 a4
a2 a5
a7 a6
a7 a5
a7 a4
a7 b6
b8 a6
b8 c6
b8 d7
e2 e3
e3 e2
```



```
LEGAL
LEGAL
ILLEGAL
LEGAL
LEGAL
ILLEGAL
ILLEGAL
LEGAL
LEGAL
ILLEGAL
LEGAL
ILLEGAL
```

IRON HACK

Forget about complex moves: en-passant, castling…

Build a Board Class

Create a 2 dimensional array as grid

Create an empty board with only two rooks

Let’s use symbols as keys in our grid

: `wR` is a white Rook

: `bR` is a black Rook

`nil` is empty space

For now, the position will be an array

Ex: [0, 0]

Later on, we will convert “a8” to [0, 0]

Add some helper function to check status of a specific position - given the position, what piece, if any, is there?

Test the methods

Next tips

Start with the Rook Class

Create a method to check whether the move is valid

The method should expect as parameters: board, origin, destination

First check for obvious bad moves.

Forget about out of the board moves by now.

Already someone on there? What color?

Check if the cells until the destination are empty

Check whether the move is horizontal or vertical

Test it!

Now you have one Piece for testing

Let’s create a new one.

Let’s create the Queen!

Copy and paste the Rook, change the name class

Add a method to check whether the movement is diagonal

Test it!

Wait! Did I just say copy and paste? That smells bad!

Create a Piece class and move the duplicated logic of Rook and Queen there

Rook and Queen should inherit from Piece

Let's create the ChessValidator class

The ChessValidator will be in charge of initializing the board

Add a method in there that converts “a1” to [0, 7]

Test it!

Create a method that takes one full move “a1 a2” and returns if it’s valid

This method should get the piece in the origin position from the board

Then we will ask the Piece if the move from origin to destination is legal with that board

Test it!

Now add a method that gets an array of moves and prints whether each one is valid or not

Test it!

Now create the rest of the Pieces and keep playing!