This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Rewrite the code from paragraph 2

Let us take the rook movement logic introduced in the previous paragraph and encapsulate it in a function called check_rook(). Inside this function, we call three helper functions, each specialized in a specific check:

- is_null_move(): checks if the move is null;
- is_on_board(): checks if the destination coordinates are within the bounds of the board;
- is_rook_move_geometry(): checks only the geometry of the rook's move, i.e., that the displacement is horizontal or vertical.

```
# Check if the move is null
def is_null_move(r_start, c_start, r_end, c_end):
     if r_start == r_end and c_start == c_end:
         return True
     else:
         return False

# Check if the destination square exists
def is_on_board(r, c):
    """Returns True if the coordinates are between 0 and 7."""
    if r >= 0 and r <= 7 and c >= 0 and c <= 7:
        return True
    else:
        return False

# "Rook specialist" function: checks only the move's geometry
def is_rook_move_geometry(r_start, c_start, r_end, c_end):
    if r_start == r_end:
        return True  # Horizontal move
    elif c_start == c_end:
        return True  # Vertical move
    else:
        return False
```

Now we can combine these "building blocks" to create the check_rook() function, which internally calls is_null_move(), is_on_board() and is_rook_move_geometry().

```
def check_rook(r_start, c_start, r_end, c_end):
     # 1. Check if the move is null
     if is_null_move(r_start,c_start,r_end,c_end) == True:
         print("Error: null move")
         return False

     # 2. Check if the destination exists on the board
     if is_on_board(r_end, c_end) == False:
         print("Error: move off the board!")
         return False

     # 3. Check if the move geometry is correct for a rook
     if is_rook_move_geometry(r_start, c_start, r_end, c_end) == True:
         print("Valid move!")
```

Co-funded by the European Union

The project "ChessAIThon" (2025-1-ES01-KA220-VET-000354329) is co-funded by the European Union. The views and opinions expressed in this publication are those of the author(s) only and do not necessarily reflect those of the European Union or the Spanish Service for the Internationalisation of Education (SEPIE). Neither the European Union nor the National Agency SEPIE can be held responsible for them.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

```
        return True
    else:
        print("Invalid move for a rook.")
        return False
```

The check_rook() function only checks for a null move, board boundaries, and the rook's move geometry (horizontal/vertical), but it does **not**verify whether there are obstacles along the path. Checking that the path is clear, which is required to fully validate a rook move, will be introduced and implemented in Paragraph 4.

The is_null_move() and is_on_board() functions can also be called inside check_knight(), thus avoiding code duplication.

```
def check_knight(r_start, c_start, r_end, c_end):
    # 1. Check if the move is null
    if is_null_move(r_start,c_start,r_end,c_end) == True:
        print("Error: null move")
        return False
    # 2. Check if the destination square exists on the board
    if is_on_board(r_end, c_end) == False:
        print("Error: move outside the board!")
        return False
    # 3.Check if the move geometry is correct for a knight
    if is_knight_move_geometry(r_start, c_start, r_end, c_end) == True:
        print("Valid move!")
        return True
    else:
        print("Invalid move for a knight.")
        return False
```

The is_knight_move_geometry() function encapsulates the logic discussed in Section 2.

```
def is_knight_move_geometry(r_start, c_start, r_end, c_end):
    delta_r = abs(r_start - r_end)
    delta_c = abs(c_start - c_end)
    if (delta_r == 2 and delta_c == 1) or (delta_r == 1 and delta_c == 2):
        return True
    else:
        return False
```

Co-funded by the European Union

The project "ChessAIThon" (2025-1-ES01-KA220-VET-000354329) is co-funded by the European Union. The views and opinions expressed in this publication are those of the author(s) only and do not necessarily reflect those of the European Union or the Spanish Service for the Internationalisation of Education (SEPIE). Neither the European Union nor the National Agency SEPIE can be held responsible for them.