Project 3: Checkers Game

Student 1: Alexis Mihaloianis
Student 2: Nita Tudor

1. General Description

- A terminal-based Checkers game implemented in C++.
- Two separate executables: Host and Client applications.
- Bidirectional flow via shared state files.

2. Functional Requirements

- Initialize an 8x8 board with pieces in starting positions.
- Host writes the initial board state to "board_state.txt".
- Client reads board state, displays board, and prompts for Player 2's move.
- Host reads Player 1's move from "board_state.txt", updates board, and rewrites the sta
- Validate moves, captures, and king promotion rules.
- Log move history to "moves_history.txt".
- Game ends when a player has no legal moves.

3. File Structure and Examples

board_state.txt – current board matrix (8x8):

```
. r . r . r . r
r . r . r . r .
. r . r . r . r
. . . . . . . .
. . . . . . . .
. b . b . b . b
b . b . b . b .
. b . b . b . b
```

moves_history.txt – recorded moves:

```
d2-c3
g7-f6
c3-e5
h6-g5
```

4. Executable Interaction

Compile both applications using:

```
g++ host.cpp -o host
g++ client.cpp -o client
```

Usage flow:

1. Host initializes board:
   ```
   ./host board_state.txt moves_history.txt
   ```
2. Client reads and plays Player 2's move:
   ```
   ./client board_state.txt moves_history.txt
   ```
3. Host reads Player 1's move and updates state.

Each executable reads from and writes to the same files to ensure bidirectional sync.