CSE 259 Spring 2026

Project 2

1. Project description

In this project, we are going to complete a prolog program that prints out a chess board so that you can watch your agents play against each other automatically! We will continue making this project a ***small team project*** (***2-3*** students per team, 3 preferable). Happy prolog!

2. Requirement specification:

A chess game with two players: playerA (white) and playerB (black). A partially implemented and runnable code is provided. To run the program, load it first (ignore the warnings, which are there for the ease of understanding) and then run it by typing:

```
| ?- main.

white move -> e2e4.                    %You are asked to provide the move for the white player
Working...

black move: e7e5, Rating: bookB        % black player's move is provided by the computer
[state(white,_350,_351,_352),state(black,_355,_356,_357),piece(a-8,black,rook),piece(b-8,black,night),piece(c-8,black,bishop),piece(d-8,black,queen),piece(e-8,black,king),piece(f-8,black,bishop),piece(g-8,black,night),piece(h-8,black,rook),piece(a-7,black,pawn),piece(b-7,black,pawn),piece(c-7,black,pawn),piece(d-7,black,pawn),piece(f-7,black,pawn),piece(g-7,black,pawn),piece(h-7,black,pawn),piece(a-1,white,rook),piece(b-1,white,night),piece(c-1,white,bishop),piece(d-1,white,queen),piece(e-1,white,king),piece(f-1,white,bishop),piece(g-1,white,night),piece(h-1,white,rook),piece(a-2,white,pawn),piece(b-2,white,pawn),piece(c-2,white,pawn),piece(d-2,white,pawn),piece(f-2,white,pawn),piece(g-2,white,pawn),piece(h-2,white,pawn),piece(e-4,white,pawn),piece(e-5,black,pawn)]

white move ->                          % You are asked again
```

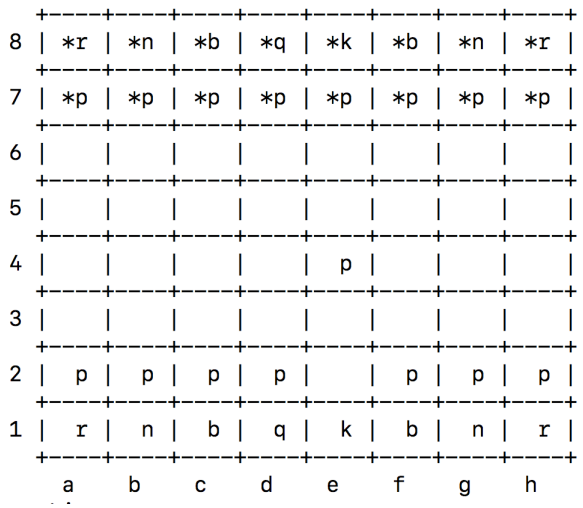
You can already play a game with the computer (if you make an illegal move, the program will exit)! There are three tasks. Completing all of these tasks will complete the code for your chess game.

2.1. Task 1: you must replace the following predicate in prolog. The current predicate only prints the data structure described below:

print_board(Board). where *Board* is a list of the following format that specifies the game state:

```
[state(white,_350,_351,_352),      // Do not worry about the numbers here which are labels for variables that
state(black,_355,_356,_357),       // keeps track of the game state; you won't need to use them in your predicate
piece(a-8,black,rook),             % The black rook is in a-8: 'a' is the File (column) and '8' is the Rank (row)
piece(b-8,black,night),            % The black night is in b-8
piece(c-8,black,bishop),           % The black bishop is in c-8
…]
```

Your predicate should convert the above list into something like the following. It should show the board and game state. Pieces with a * represent black pieces.



Hint: you can check whether a piece is in the current game state by using "member(piece(File-Rank, Color, Type), Board).

2.2. Task 2: Implement playerA's code.

Hint: very similar to playerB's code. The challenge here is to understand playerB's code and adapt it for playerA.

2.3. Task 3: Use playerA's code to play against playerB.

Hint: no need to write much code here. The challenge is to understand the main process of the chess program.

3. Submission:

Submission is ***electronically*** via canvas in a zip file. ***One and only one*** member must submit the file. It should contain the following files:

a. README.txt: this file should include names of you team members ***and*** each of your contributions; be precise
b. chess.pl: your code; make sure to test it thoroughly and ***comment*** properly.

4. Grading:

Grading will be based on the following criteria:

a. Task 1: 35%
b. Task 2: 35%
c. Task 3: 20%
d. Comment: 10%