Name: Du Tham Lieu

Student ID: 18BS615

## Prompt:

You have been asked to create a chess-playing system for tournament play. The system is built into a normal chessboard and uses a camera to view the positions on the board, it then makes its moves by declaration on a screen (an official of the tournament will move the pieces for it), it will also be able to operate the clock, and must be able to hand in its scorecard.

## Task: Provide a PEAS model for the system.

**Agent type**

Chess-playing computer.

**Performance measure**

Assuming the system joined the tournament as an individual player, the performance measurement would be its total score after a tournament.

**Environment**

The chessboard, the tournament

**Actuators**

1. Man, an official of the tournament, to move the chess pieces
2. The prompt is vague on purpose, it is unclear whether an official of the tournament would press the clock after making a move or the system/computer (like deep blue vs Kasparov match in 1997 [1]) will have a mechanism to interact with the clock mechanically/wirelessly. In the first case, most probably, the man is also an actuator for the system to "operate the clock". In the second case, the actuator would be the clock

Name: Du Tham Lieu

Student ID: 18BS615

itself as a part of the system that only takes input from whomever the machine is playing against. i.e pressing the clock

1. Resign
2. Screen to display scorecard, a possible implementation of “hand in its scorecard” is to have it digitally displayed on its screen

**Sensors**

1. The chess-playing system uses a camera to view the positions on the chessboard

## Task: Define the task environment. Provide a clear justification for these choices.

The task of the system: To win or to be precise, not lose, as many games as possible in the tournament

1. **Fully Observable**, the environment is said to be fully observable if the agent can see everything in the world via its sensor. For our chess-playing system, its world is the chessboard and the moves that its opponent makes, all of which can be seen via its camera. Therefore, the environment is said to be fully observable

2. **Multiple Agents**. Even though the computer can play against itself, chess is usually played with another agent. Furthermore, in the prompt, the chess play system is created for a tournament. Therefore, the environment is multi-agent.

Name: Du Tham Lieu

Student ID: 18BS615

3. **Deterministic**. Chess is a game with well-defined rules, at any stage of the game, all pieces have a set of moves they can take, these moves are finite and can be determined. Therefore, the environment is deterministic.

4. **Sequential**. In the game of chess, each player taking turns one after another, each move the players take will influence their next move. Therefore, the environment is sequential

5. **Dynamic**. Every move each player take can drastically affect the outcome of the game. Therefore, the environment is dynamic.

6. **Discrete**. Due to the nature of the game of chess, the environment is discrete because it plays one move at a time. Therefore, the environment is discrete.

7. **Known**. The designer of the chess-playing system has full knowledge of the rules and the composition of the board.

## Task: Compare and contrast two methods for the system to find the next move. Comment on these two methods in regards to how they use Exploration and Exploitation.

**First method**

First of all, for the sake of the argument, let's assume our chess-playing AI has a basic "understanding" of chess rules, or at least it wouldn't make an illegal move and we do not have access to any opening/endgame library. One can start by is to use an algorithm to find the best move for each ply (one ply is half a turn, in chess terminology a turn means each player make one move), one of the most simple ways a chess engine can search for the next best move is for each ply calculate the best outcomes by the use of a decision tree [1]. In order for this to work, a

Name: Du Tham Lieu

Student ID: 18BS615

value (or weight) is associated with each piece. For example, a pawn worths 1 point, a knight worths 2.5 points, a rook worth 5 points, and a Queen worth 10 points, etc... In each move, the system iterate overall outcomes exhaustively depth-first on each branch of said tree to minimize loss (or maximize gain by capturing and trading pieces) for a predetermined depth (for example 3) then wait for the opponent to finish their ply and repeat until the game is concluded.

**Comment on the first method**

This method relies heavily on exploitation. Although the method explores the possibilities of a better outcome over a decision tree but in practice, such a system that implements this algorithm usually takes a greedy approach to accomplish more with limited resources. In another word, a whole branch of the game/decision tree is cut off if the sibling node yields a worse value than the first node after the first branch had been evaluated.

**Second method**

Continuing with the game tree example above, perhaps, the best moves could be more efficiently found at each level. One can change their approach to do an essentially, breadth-first search (BFS ) instead of the depth-first search (DFS) mentioned above. Such an approach has many implementations, the “best-fit” algorithm is one such candidate. The aim of such an approach can be summarized nicely (by) “expand the most promising node of one level first. Best-first turns an uninformed breadth-first into an informed search.”[3] (explanation)

**Comment on the second method**

This approach faces the same drawbacks as the first approach. The only thing that differs is BFS method leans more on the exploration side, assuming that the agent the system is playing against, together with the system itself does not gain any benefit from altering their approach.

Name: Du Tham Lieu

Student ID: 18BS615

## Task: Can this system ever be said to know how to play chess? Why or why not?

“Can this system ever be said to know how to play chess?” Before considering this question, it is important to point out that despite the two methods for searching the “next-best-move” described above is somewhat primitive at best if implemented correctly, it could potentially beat the average beginner human chess player. It is not entirely wrong to assert that the fictional system, regardless of how poorly it may perform, “can play some chess”. Given the proposition “can play chess is the same as knowing how to play chess ” is true, then the system can certainly be said to know how to play chess, even at a beginner level.

## Footnotes:

[1] “Deep Blue versus Garry Kasparov,” *Wikipedia*, 27-Feb-2021. [Online]. Available: https://en.wikipedia.org/wiki/Deep\_Blue\_versus\_Garry\_Kasparov. [Accessed: 04-Mar-2021].

[2] freeCodeCamp.org, “A step-by-step guide to building a simple chess AI,” *freeCodeCamp.org*, 15-Mar-2017. [Online]. Available: https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/. [Accessed: 04-Mar-2021].

[3] “Best-First,” Best-First - Chessprogramming wiki. [Online]. Available: https://www.chessprogramming.org/Best-First. [Accessed: 04-Mar-2021].