Computer Science 5400
Artificial Intelligence

Spring 2022

Game Assignment #1

Version 04.01

DUE DATE : Monday, April 18th, 11:59:59pm

Assignment:

Create a program that makes **random**, but **valid**, chess moves over the provided chess **AI Framework**.

Objectives:

The main objective of this assignment is for you to create your own model of the game state and action generation to be used in the assignments of this series.
In brief:

- **generate all valid moves for each of your (black/white) player's pieces in the current game state.**
- randomly select a piece and a valid move to make.
- model the results of performing the selected move in your own state representation.

WARNING: This is an individual project; plagiarism of any type will result in a grade of zero (0). To be clear, anyone who even imports/includes chess libraries that they did not personally write, will instantly get a zero (0).

Specifications:

Build your program over the provided **AI Framework**. You will need to fill in the `make_move()` function of the AI class in the provided AI framework template to implement an AI player for Chess in either **C++**, **C#**, **Java**, or **Python3**.

Output:

On each turn, output the move chosen by your AI program in **Long Algebraic Chess Notation** [https://en.wikipedia.org/wiki/Algebraic_notation_(chess)] in the **Universal Chess Interface** [https://en.wikipedia.org/wiki/Universal_Chess_Interface] format. with any necessary additional information needed to describe the move. Your program may need to translate the starting and ending square of the move into standard chess notation in order to output in the correct format. For this first game assignment only, your program shall also print out **all moves** associated with the piece that is to be moved on each turn of the game.

Documentation:

Further documentation regarding the **AI-framework** is available at the following link, [~~http://docs.siggame.io/chess~~]

- C++: [http://siggame.github.io/Joueur.cpp/namespacecpp__client_1_1chess.html]
- C# : [http://siggame.github.io/Joueur.cs/games/Joueur.cs.Games.Chess.html]
- Python : [http://siggame.github.io/Joueur.py/chess/index.html]
- Java : [http://siggame.github.io/Joueur.java/#] (Packages > games.chess)

Submission:

Place all code/scripts in your git repository in the course's GitLab server, [link]
(If you still do not have a repo by Friday, let a TA know).
Upon your final submission, your repository must contain the following:

- **Source code**, along with any additional dependencies your program needs in order to be built, but **not** the server.

- "`ReadyToSubmit.txt`" text file with the content:
    - "`Yes`" or "`yes`" on the **first** line
    - The language you are using on the **second** line: "`cpp`", "`py`", "`cs`", or "`java`".

Leaving the first line of "`ReadyToSubmit.txt`" as "`No`" indicates you are not ready (i.e. if you intend to submit late). Once you have changed the file to "`Yes`"/"`yes`" and have indicated your language of choice, your assignment will be considered submitted.

Grading:

Your code will be tested and graded with the "`play.sh`" script, which will play your AI against itself, and executed as in:

```
./play.sh LANGUAGE_DIR SESSION_ID
```

Where `LANGUAGE DIR` is the directory you are coding in (e.g. `Joueur.py`), and `SESSION_ID` is the session ID for your game.

NOTE: By default the "`testRun`" script, in your respective language directory, is set up to connect to the remote host (`cashdomain.tk`). If you wish to run the game server locally, you will need to switch this to connect to localhost (`127.0.0.1`). You can download the server software from https://github.com/siggame/Cerveau (SUBNOTE: Ignore the link that is provided for log files, you can find the logs under `logs/gamelogs` in the server directory, if you are running the server locally).

More on Grading:

Your program should need no manual configuration by the TAs to compile and execute your program; this should be accomplished by simply running the provided bash script ("`play.sh`"). **DO NOT EDIT** "`play.sh`".
Your program will be evaluated and graded on the Computer Science department's Linux machines so your program needs to be compatible with the current system,

compilers and environment.

Grading Rubric:

Description	Weight
Correct Action and State Generation	70%
Search Algorithm (Random Search)	10%
Good Programming Practices (adherence to coding standards, modular design, documentation etc.)	20%

You are required to follow good programming practices, such as [Dr. T's Coding Standards and Tips].
You are also expected to properly document your code.