SMAI: Assignment 3

Trial Submission: November 7
Final Submission: November 21

Problem Statement: To code a bot to play the game of Othello

https://www.wikihow.com/Play-Othello

https://en.wikipedia.org/wiki/Reversi#Othello

Description: In this assignment, you will code a bot to win the game of Othello. Given a board configuration and a turn, your bot will return a valid move. The game ends when neither of the players can make a valid move. The player with a larger number of coins when the game ends is the winner.

Programming Language: C++
System specifications (preferred): 64-bit Linux distribution.

Instructions:

Setting up the framework

We will be providing you with a framework (Desdemona) that lets two bots compete against each other.

1. Extract the contents of Desdemona into a suitable directory.
2. Set up the framework by issuing a make command in the root of this directory.

Coding the bot

- You will modify MyBot.cpp to return a valid move whenever the function play is called. The source is located at bots/MyBot.
- All other source files are to be left untouched.
- The makefile is also provided at this location. Use it to generate a .so file.
- You can test your bot against another bot by issuing the command "./bin/Desdemona ./<path to bot1.so> ./<path to bot2.so>"
- By convention, the first bot is BLACK and the second RED.
- A random bot (bots/RandomBot) has been provided for testing.
- At the end of the game, a game.log file is created that contains the sequence of moves made.
- The bots being submitted must have NO print statements.
- If a bot returns an invalid move, it will be disqualified.

Helper functions

The following functions have already been written to assist you:

- bool OthelloBoard::validateMove(Turn turn, int x, int y)
  true if the passed move (x,y) is valid for the passed turn, false otherwise
- bool OthelloBoard::validateMove(Turn turn, Move move)
  true if the passed move is valid for the passed turn, false otherwise
- void OthelloBoard::makeMove(Turn turn, int x, int y)
  Updates the board configuration by making the move (x,y); throws an exception if the move is not valid
- void OthelloBoard::makeMove(Turn turn, Move move)
  Updates the board configuration by making the specified move; throws an exception if the move is not valid
- list¡Move¿ OthelloBoard::getValidMoves(Turn turn)
  Returns a list of valid moves that can be made given the turn
- int OthelloBoard::getBlackCount()
  Returns the number of black coins on the board
- int OthelloBoard::getRedCount()
  Returns the number of red coins on the board
- void OthelloBoard::print(Turn turn)
  Prints the turn, the board configuration, and the number of black and red coins. ’X’ is BLACK, ’O’ is RED, and unfilled locations are blank

Time Constraints

Each bot can take at most 2 seconds to return a move. If this time limit is exceeded, the bot causing the timeout will be disqualified.

Evaluation

- The assignment is to be done in teams of two.
- A complete round robin tournament of all bots will be run. The outcome will determine the marks for the assignment.
- If running a complete tournament is not possible then each bot will be pitted against 5 other randomly chosen bots and the results will be translated to assignment marks.

Submissions

Trial Round
Upload a single .so file with the enrolment no .so (e.g.: u20210008.so).

Final Round
Upload a zip file containing your source code, writeup and the enrolment no.so file. The zip file is to be named enrolment no.zip (e.g. u20210008.zip)