CS4200 -- Project 3: Game (4-in-a-line)

You are only allowed to use Java, C++

The Game:
You have a 8x8 board, players take turns placing a piece on any grid. First player to get 4 in a line (either a row, or a column; diagonals are NOT counted) wins. The maximum amount of time allowed for generating the next move is 25 seconds. At the very beginning, your program should ask the user for the maximum amount time (in seconds) allowed for the computer to generate the answer and make sure that your search will stop when there's no time left. Your program should also ask the user to decide who is going to move first.
In order to make the process smooth, you are **REQUIRED** to use the following standard for displaying the movement that your program will generate:

```
  1 2 3 4 5 6 7 8      Player vs. Opponent
A - - - - - - - -         1. e5 d5
B - - O - - - - -         2. e4 e3
C - - O - - - - -         3. f4 d4
D - X O O O X - -         4. e6 e7
E - - O X X X O -         5. g4 h4
F - - X X - - - -         6. d6 d3
G - - - X - - - -         7. d2 c3
H - - - O - - - -         8. f3 b3 wins
```

Strategies and Other Requirements

1. To get a full credit, your program must satisfy the requirements listed above, and MUST use alpha-beta pruning to determine the computer's move. Note that, you will need to make some changes to the alpha-beta pruning introduced in the book. These changes include: an evaluation function so that you can evaluate non-terminal states and a cut-off test to replace the terminal test so that your program will return the best solution found so far given a specific period of time.

2. In general, you are allowed 25 seconds to run your algorithm, which should search at least 5 plies deep. If you use languages other than C or C++, the number of plies may be smaller. You can implement iterative deepening search with alpha-beta pruning so that your program will search as deep as possible given the fixed time limit.

3. Given the same amount of time, if your terminal evaluation function is simple, then your program should be able to search deeper than one with a complex (but maybe better) terminal evaluation function. You can decide whether you want a deeper search and a simple evaluation, or a shallower search with a complex evaluation.

4. Your program is one player, and will attempt to defeat the opponent.

5. Your program must prompt the user for who goes first; player or opponent.

6. The program should detect when an illegal move has been entered and requires the human to re-enter a valid move, for example, trying to place on a non-empty space, or out of bounds

Note: A group of 2 is allowed to work on this project. You can also work individually. Your final executable program should be able to run on the computers in the CS labs.

How do two programs interact?
To make a common interface, your program should output the current choice for each step. Each program will run on one computer and the final result will be the opposite of each other. Please see the following illustration:

The program running on computer 1 goes first: the program should search for its first move, for example, "e5"

```
        1 2 3 4 5 6 7 8     Player vs. Opponent
      A - - - - - - - -        1. e5
      B - - - - - - - -
      C - - - - - - - -
      D - - - - X - - -
      E - - - - - - - -
      F - - - - - - - -
      G - - - - - - - -
      H - - - - - - - -

        Player’s move is: e5

       Choose Opponent’s next move:
```

<u>Program running on computer 2</u>: this program is waiting for the first player’s move (opponent) to be entered. After which, this computer will search for the best responding move and draw the resulting board.

```
    Choose Opponent’s next move: e5

        1 2 3 4 5 6 7 8     Opponent vs. Player
      A - - - - - - - -       1. e5  d5
      B - - - - - - - -
      C - - - - - - - -
      D - - - - 0 - - -
      E - - - - X - - -
      F - - - - - - - -
      G - - - - - - - -
      H - - - - - - - -

        Player’s move is: d5

        Choose Opponent’s next move:
```

This process will continue until the end of the game. The program must announce a win, lose, or draw then terminate. If the opponent goes first the order in which the label and move log appears will have to be changed; e.g., “Opponent vs. Player” or “Player vs. Opponent”

Requirements:

- Your computer is required to prompt for “Who goes first, C for computer, O for opponent: ”
- Your computer will be required to make its move within 25 seconds.
- The board will be 8 x 8 with the coordinate A1 indicating the top left hand side of the board.
- The user interface must display the game as demonstrated above
- When a new game starts, It must prompt for the time limit per move and prompt for who goes first (computer or opponent)
- Moves must be entered as shown above
- Your game must inject some randomness into its moves.
    - If there are multiple optimal moves that result in the same evaluation value, it must randomly choose from those moves, this only needs to happen for the initial moves of the game.
- **The game must be implemented using the minimax algorithm with alpha-beta pruning**
    - It will help if you also implement this with an iterative deepening.

What to Submit and How to Submit it:

- You can work in teams of no more than two members. Both team members must submit a copy of the the project. Teams must be reported by the assigned due date (to be announced). After the reporting due date, teams cannot be formed
    - You can also work individually if you want
- Source code + README (how to compile and run your code). Both team members can submit the same README.
    - Do not assume that the grader will use your IDE
    - You need to instruct the grader on how to compile and run your program from the command line
- **Each team member should write their own report** on the strategies that were employed in creating the evaluation function. Discuss the problems that you encountered and the steps that you took to resolve them. The report should be <= 3 pages.
- Create a folder called, "lastname_firstname_4200p3", that includes all of the required files, from which, you should generate a zip file called "lastname_firstname_4200p3.zip". For example, if Jane Doe is submitting a project, she would name the folder doe_jane_4200p3. The resulting zip file would be named, doe_jane_4200p3.zip. Submit this file via Blackboard before the due date.

No late submissions will be accepted

The tournament:
On the last day of class we will hold a competition. Teams will be arranged into groups of 2 teams. The teams in each group will play against each other. Each match contains three games, with a coin toss to determine who goes first (BYOC). The best of three games wins the match. The next round will consist of winning teams from the previous round. This process will continue until there is one team standing. **The member(s) of the first-place team will receive 5 points added to their final grade. The runner-up will receive 3 points**. If a program crashes during play, the team that crashed will lose that game. If a team's play exceeds the allotted time (25 seconds), the team that exceeded the time limit will lose that game.

Grading:

Minimax/alpha-beta implementation Including the evaluation function	50%
User interface	15%
Code quality	15%
Report and README	15%
Participation in the competition	5%

Note: All team members must be present. Only those teams with all members present will get 5% for attendance
You will be graded on your submitted code, If your program does not compile and run, you will not receive any points for the project. If you manage to win the tournament but your submitted code is not competent, you will forfeit the extra credit. You must use your submitted code at the tournament.