Game: Lines of action
Board size: 6x6, 8x8

Android app: https://play.google.com/store/apps/details?id=teddydroid.app.loa.free (if you want to play)

- You can disallow location permission, it will work perfectly in most cases.
- Rules of the game can also be found in this app.

iOS app: https://apps.apple.com/us/app/lines-of-action/id503451098

Play on web:

- http://www.ludoteka.com/juegos?juego=lineas-de-accion
- https://en.boardgamearena.com/gamepanel?game=linesofaction

**Rules of the game:**
http://www.boardspace.net/loa/english/index.html#howto-play (also check the unusual endgames)
https://en.wikipedia.org/wiki/Lines_of_Action
(We are considering no draw)
http://www.iggamecenter.com/info/en/loa.html (consider red checkers as black here)

**Heuristics:**
To be added by Wasif sir

You have to design UI and write move logic using adversarial search with alpha-beta pruning. When the player clicks on a specific checker, show its all possible legal positions after move (or show “no legal move” if there isn’t any for this checker).

Keep provision to play

- Human vs human
- Human vs AI

Each move shouldn’t take more than 1 second in 6x6 board and 2 seconds in 8x8 board (considering move generation time only). At each depth, keep a solution ready based on heuristic. When the time is over, return the current best solution and make a move according to it.

- Keep your code clean and modular. Don’t write the same block of code multiple times, rather use functions.
- You can use any language (C++, Java, Python). But make sure to make the code efficient. Otherwise you may need to prune at a very low depth, which will result in poor performance of the game agent. No need to mention, Python is inherently slow in such searches unless you make the algorithm very efficient.

- You can implement minimax algorithm in one programming language and UI in another programming language. In that case, you may need an intermediary common language such as JSON.
- Copy all your files & folders in a single folder and name that folder with your 7 digit student id (say, 1605125). Then compress that folder to a zipped file and submit that zipped file to moodle.

**Mark distribution:**
Minimax algorithm with alpha beta pruning (with proper heuristic) - 5
Designing a UI that properly collaborates with minimax algorithm - 3
Code cleanness, modularity & efficiency - 1
Proper submission - 1

**Submission deadline**:
7 November 2020 11:55 pm