CSE 4301/5290 Homework 3

Due: Oct 22, Wed, 5pm; Submit Server: course = ai , project = hw3

1. Q5.9, p197, 3Ed (Q6.1, p189, 2Ed)
2. Programming (LISP, Java, C, C++, or Python): *HOP* has a 6×6 board and each player has 6 pieces. The pieces are initially placed on the first row on opposite sides. The objective is to move all the pieces to the last two rows ("end zone") on the other side.

```
   0 1 2 3 4 5
  -------------
0 |X|X|X|X|X|X|
  -------------
1 | | | | | | |
  -------------
2 | | | | | | |
  -------------
3 | | | | | | |
  -------------
4 | | | | | | |
  -------------
5 |O|O|O|O|O|O|
  -------------
```

   Players alternate to move their pieces. At each turn:

   (a) A piece can be moved to any adjacent empty square (including diagonal ones).

   (b) A piece can hop over another piece to another empty square. If the distance between the current piece and the "booster" piece is n empty squares, the current piece will land n empty squares away from the "booster" piece in the same direction (including diagonal ones). The "booster" piece can be your own or your opponent's.
   - **CSE 4301**: at most one hop is allowed in one move
   - **CSE 5290**: multiple hops are allowed in one move (different hops can be in different directions)

   (c) Moving to adjacent square and hopping cannot be combined in one move.

Program operations:

   (a) From keyboard input, assign your program to be player X or player O (the opponent respectively becomes player O or X).

   (b) Player X always starts.

   (c) Display the initial board (X begins from the top row, and O from the bottom row)

   (d) Display a move (including intermediate hops) or enter a move from the keyboard (row and column)

   (e) Check if the move is legal, **ask human** for confirmation if illegal

   (f) Make the move and display the board

   (g) Repeat steps (d) to (f) until there is a winner

   (h) Declare winner

Program requirements:

   (a) Functions (stated in LISP) including:

```
; perform search with alpha beta pruning, return an action
; *all* actions must be determined by alpha-beta-search
; (you can vary parameters at different states)
(defun alpha-beta-search (state ...)  ...)

; return the "quality" of the state
; description of your evaluation function...
(defun eval-state (state)  ...)
```

   (b) Each move should not take more than one minute. Hence, you might want to have a parameter(s) that sets the limit(s) of your search.

LISP details

- `compile-file` prepares a compiled version of your program. You need to `load` the compiled version. Running the compiled version will be faster than interpreting the source.
- `get-universal-time` returns the number of seconds since Jan 1, 1970. `get-internal-real-time` returns the number of time units based on `internal-time-units-per-second` (a constant)—might need to handle the "wrap-around" issue: end-time < start-time.

Tournament rules:

   (a) Oct 22, Wed, 5-6:15pm

   (b) CSE 4301 and CSE 5290 students compete separately.

   (c) Your program will play against two other programs, which are randomly assigned.

   (d) Your program starts in one game; your opponent starts in the other game.

   (e) You can get up to 10 points, which constitute 10% of your hw3 grade. (win: 5 points; tie: 3.5; lose: 2; non-functioning: 0)

   (f) Your program wins if it functions but your opponent's doesn't.

   (g) Each move cannot take more than one minute. Your program is considered non-functioning if it takes too long.

   (h) Your program is considered non-functioning if it suggests or allows illegal moves.

   (i) If a game takes more than half an hour, the number of pieces in the "end zone" is counted and the player with more pieces wins.

CSE 5290 only

3. Q5.16, p200, 3Ed