Assignment 1: Game Search

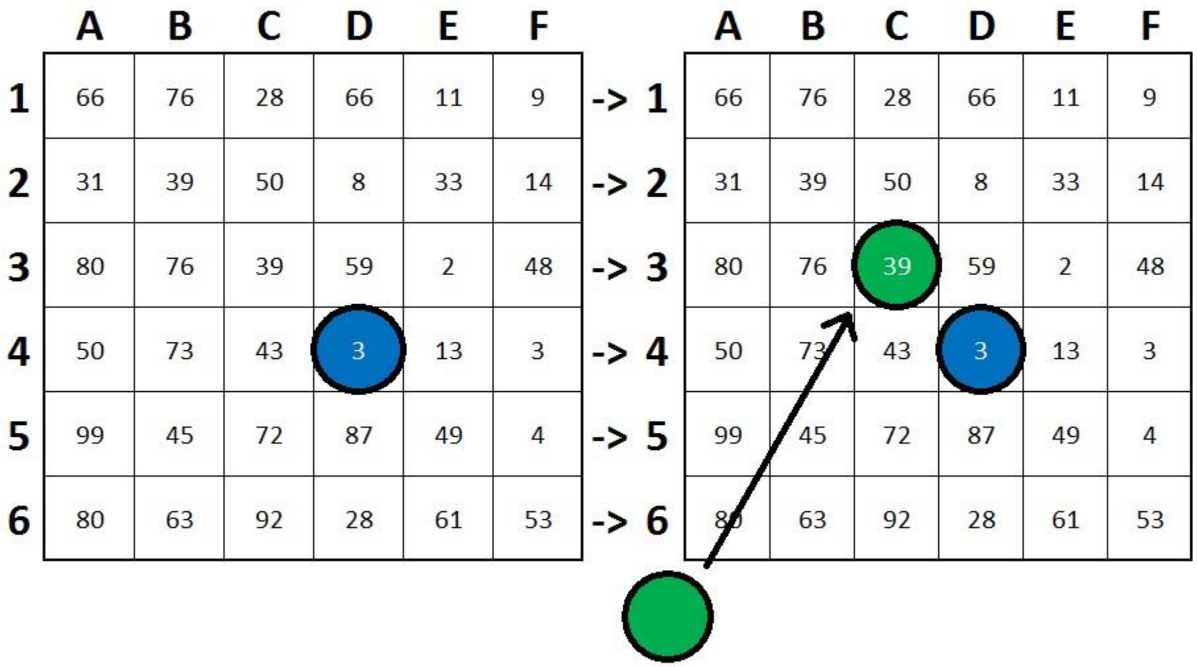
The goal of this part of the assignment is to implement an agent to play a simple "warfare" game.

Rules of the game

- The game board is a 6x6 grid representing a city.
- Each square has a fixed point value between 1 and 99.
- There are two players, "blue" and "green". Each player takes turns: blue moves first, then green, then blue, etc.
- The object of the game is to be the player in the end with the largest total value of squares in their possession. That is, one wants to capture the squares worth the most points.
- The game ends when all the squares are occupied by all players since no more moves are left.
- Movement is always vertical and horizontal but never diagonal.
- Pieces can be conquered in the vertical and horizontal direction, but never the diagonal direction.
- The values of the squares can be changed for each game, but remain constant within a game.
- In each turn, a player can make one of two moves:

  Commando Para Drop: You can take any open space on the board with a Para Drop. This will create a new piece on the board. This move can be made as many times as one wants to during the game, but only once per turn. A Commando Para Drop cannot conquer any pieces. It simply allows one to arbitrarily place a piece on any unoccupied square on the board. Once you have done a Para Drop, your turn is complete.

  The image below illustrates a Commando Para Drop. In this case, green drops a new piece on square [C,3]. This square is worth 39, which is a higher number, meaning that it contains some juicy oil wells or other important resources. After that, the score is green 39 : blue 3. A Commando Para Drop could have been carried out on any squares except for [D,4] since blue already occupies it.

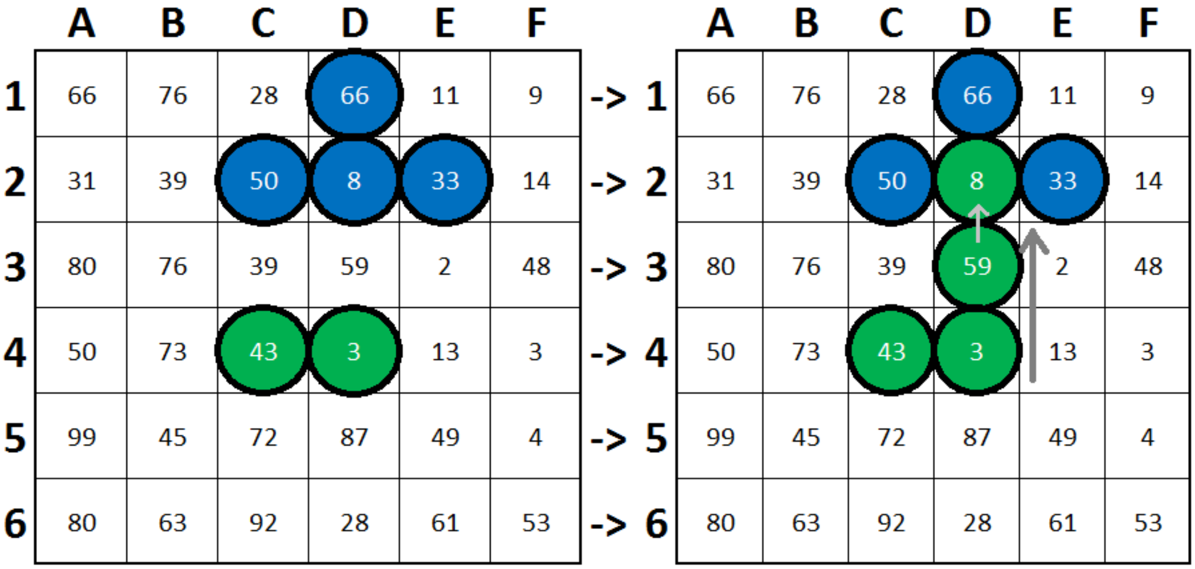


	A	B	C	D	E	F
1	66	76	28	66	11	9
2	31	39	50	8	33	14
3	80	76	39	59	2	48
4	50	73	43	3	13	3
5	99	45	72	87	49	4
6	80	63	92	28	61	53

	A	B	C	D	E	F
-> 1	66	76	28	66	11	9
-> 2	31	39	50	8	33	14
-> 3	80	76	39	59	2	48
-> 4	50	73	43	3	13	3
-> 5	99	45	72	87	49	4
-> 6	80	63	92	28	61	53

  M1 Death Blitz: From any space you occupy on the board, you can take the one next to it (up, down, left, right, but not diagonally) if it is unoccupied. The space you originally held is still occupied. Thus, you get to create a new piece in the blitzed square. Any enemy touching the square you have taken is conquered and that square is turned to your side (you turn its piece to your side). An M1 Death Blitz can be done even if it will not conquer another piece. Once you have made this move, your turn is over.

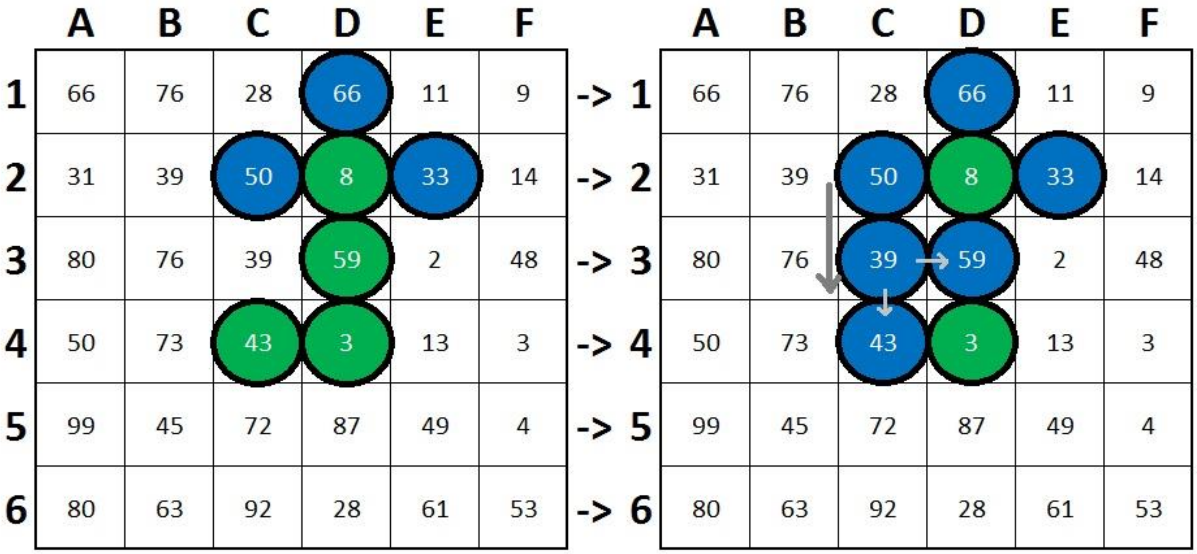
The image below illustrates an M1 Death Blitz. Green blitzes the piece in [D,4] to [D,3]. This conquers the blue piece in [D,2] since it is touching the new green piece in [D,3]. A blitz always creates a new piece and always moves one square, but it does not conquer another piece unless it is touching it. Thus, another valid move might have been for [D,4] to have blitzed [E,4]. Then the green player would own [D,4] and [E,4] but would have conquered none of blue's pieces. Note, the score before the blitz was green 46 : blue 157 but afterwards is green 113 : blue 149.



	A	B	C	D	E	F
1	66	76	28	🔵 66	11	9
2	31	39	🔵 50	🔵 8	🔵 33	14
3	80	76	39	59	2	48
4	50	73	🟢 43	🟢 3	13	3
5	99	45	72	87	49	4
6	80	63	92	28	61	53

	A	B	C	D	E	F
-> 1	66	76	28	🔵 66	11	9
-> 2	31	39	🔵 50	🟢 8	🔵 33	14
-> 3	80	76	39	🟢 59	2	48
-> 4	50	73	🟢 43	🟢 3	13	3
-> 5	99	45	72	87	49	4
-> 6	80	63	92	28	61	53

Here is another illustration:



	A	B	C	D	E	F
1	66	76	28	🔵 66	11	9
2	31	39	🔵 50	🟢 8	🔵 33	14
3	80	76	39	🟢 59	2	48
4	50	73	🟢 43	🟢 3	13	3
5	99	45	72	87	49	4
6	80	63	92	28	61	53

	A	B	C	D	E	F
-> 1	66	76	28	🔵 66	11	9
-> 2	31	39	🔵 50	🟢 8	🔵 33	14
-> 3	80	76	🔵 39	🔵 59	2	48
-> 4	50	73	🔵 43	🟢 3	13	3
-> 5	99	45	72	87	49	4
-> 6	80	63	92	28	61	53

Here blue blitzes [C,3] from [C,2]. In the process green's pieces at [D,3] and [C,4] are conquered since they touch [C,3]. Notice that in its next move, green will not be able to conquer any of blue's pieces and only the piece at [D,4] would be able to execute an M1 Death Blitz since [D,2] has no neighboring unoccupied squares.

You can assume that blitzing is mandatory: i.e., if you put a piece adjacent to pieces you already own and you have the opportunity to "convert" neighboring enemy pieces, you have to take it.

Your task is to implement agents to play the above game, one using **minimax search** and one using **alpha-beta search**. Your program should use depth-limited search with an evaluation function -- which you, of course, need to design yourself and explain in the report. Try to determine the maximum depth to which it is feasible for you to do the search (for alpha-beta pruning, this depth should be larger than for minimax). The worst-case number of leaf nodes for a tree with a depth of three in this game is roughly 42,840. Thus, you should at least be able to do minimax search to a depth of three.

For each of the five test game boards given as attachment, run the following matchups:

1. Minimax vs. minimax;
2. Alpha-beta vs. alpha-beta;
3. Minimax vs. alpha-beta (minimax goes first);
4. Alpha-beta vs. minimax (alpha-beta goes first).

For each matchup, report the following:

1. The sequence of moves (e.g., "blue: paradrop C3, green: paradrop A1," etc.), the final state of the board (who owns each square) and the total scores for each player;
2. The total number of game tree nodes expanded by each player in the course of the game;
3. The average number of nodes expanded per move and the average amount of time to make a move.

For bonus points: Design an interface for the game that would allow you to play against the computer.

Deliverables:

1. Programs as a zip file.
2. Output for each game as mentioned above. As a zip file.
3. A writeup.
4. 4. If GUI is built: a demo