The task of this assignment is to implement a game player which uses adversarial search algorithms to play a two player game. The basic implementation includes minimax algorithm with alpha-beta pruning. Also, you can experiment with different search strategies. You can experiment with iterative-deepening search with varying depth-limits and move-ordering.

A student should also implement various heuristics and find out which one is better by running experiments. For example, you can determine the win-loss ratio by running 100 games with computer-vs-computer autoplay.

For section A1/B1, the task is to implement a game player for othello or reversi. (http://www.othelloonline.org/ provides an online implementation.)

For section A2/B2, the task is to implement a game player for mancala. (http://play-mancala.com/ provides an online implementation.)

A heuristic functions estimates how good a particular state is for a player. A number of factors determine whether a given state of the game is good for a player.

Othello heuristic: For Othello, strategic elements such as mobility, stability, corners and parity determine how favorable a particular position is for a player.

Othello heuristic-1: (Positional heuristic) Assign static weights associated to each board position such as shown below. The idea is to maximize its own valuable positions (such as corners and edges) while minimizing its opponent's valuable positions.

	A	B	C	D	E	F	G	H
1	4	-3	2	2	2	2	-3	4
2	-3	-4	-1	-1	-1	-1	-4	-3
3	2	-1	1	0	0	1	-1	2
4	2	-1	0	1	1	0	-1	2
5	2	-1	0	1	1	0	-1	2
6	2	-1	1	0	0	1	-1	2
7	-3	-4	-1	-1	-1	-1	-4	-3
8	4	-3	2	2	2	2	-3	4

Evaluation Function:

Wa1 * Va1 + Wa2 * Va2 + ... + Wa8 * Va8 + ... + Wh8 * Vh8

Here W is +1, -1 or 0 if the square is occupied by player, opponent or empty. The V's are obtained from a positional value table such as shown above. There can be alternative schemes with which you can experiment. For example, an alternative positional value table is shown below.

	A	B	C	D	E	F	G	H
1	100	-20	10	5	5	10	-20	100
2	-20	-50	-2	-2	-2	-2	-50	-20
3	10	-2	-1	-1	-1	-1	-2	10
4	5	-2	-1	-1	-1	-1	-2	5
5	5	-2	-1	-1	-1	-1	-2	5
6	10	-2	-1	-1	-1	-1	-2	10
7	-20	-50	-2	-2	-2	-2	-50	-20
8	100	-20	10	5	5	10	-20	100

Othello heuristic-2: (Mobility heuristic)
Mobility refers to the number of legal moves a player can make in a particular position. The idea is to maximize own mobility and minimize its opponent's mobility. Corner square are important in mobility.

Evaluation Function:

$W1 * (C_{player} - C_{opponent}) + W2 * (m_{player} - m_{opponent}) / (m_{player} + m_{opponent})$

Here, c is corner squares, and m is the mobility

W1 = 10, W2 = 1

You can experiment with values of W1 and W2

Othello heuristic-3: (Absolute count heuristic)
The idea is to maximize the number of one's own disks and minimize the number of the opponent player's disks.

Evaluation Function:
$n_{player} - n_{opponent}$

Here , n is the number of disks

Mancala heuristics:

In Mancala, the board consists of six (6) bins on each side, and a home position (called storage) on the right of the bins. The board is laid out typically starting with four stones in each bin and both storage bins empty.

For Mancala, the following strategic factors can be used to design a good heuristic to determine how favorable a particular position is for a player:

(1) First valid move [furthest valid bin (a bin on my side which is not empty) from my storage]
(2) How far ahead of my opponent I am now [the difference in stone count between my storage and opponent's storage]
(3) How close I am to winning (If my storage is already close to containing half of total number of stones)
(4) How close opponent is to winning
(5) the total number of stones residing in the six bins of my side
(6) the total number of stones residing in the six bins of opponent's side
(7) Number of stones close to my storage (a stone, although residing in a bin on my side, is not close to my storage, if it is going to be overflowed to my opponent's side)
(8) Number of stones close to my opponent's storage
(9) Have I earned an extra move
(10) Have I captured any stone

You can experiment with the following four heuristics (and some of your own heuristics):

heuristic-1: The evaluation function is
(stones_in_my_storage – stones_in_opponents_storage)

heuristic-2: The evaluation function is
W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side)

heuristic-3: The evaluation function is
W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side) + W3 * (additional_move_earned)

heuristic-4: The evaluation function is
W1 * (stones_in_my_storage – stones_in_opponents_storage) + W2 * (stones_on_my_side – stones_on_opponents_side) + W3 * (additional_move_earned) + W4 * (stones_captured)