In this project, the task that we had to complete was to create a adversarial Search AI to play the game Isolation. The algorithm that we use needed to implement is the MiniMax and optimized version of Minimax, which is alpha Beta prunning.

The Rules to Isolation goes as follows; the Game is played on a square Grid of any size, in this project the Board size is seven by seven (49 places). Each player takes turns moving and is only alowed to move in a 'L' shape (2 spaces then 1 space adjacent to the directed you went, just like a Knight in chess). Once a space has been visited by either player it cannot be visited again. The objective of the game is to make your opponent not have any more moves when its their turn.

The minimax algorithm simulates the game by going through every board state from a initial board state. From a Emtpy seven by seven board there would be roughtly 49!, which is $6.08x10^{62}$ board states. Going through each state to find the most optimal move would not be efficient. To combat this problem the algorithm only goes a few levels deep from the current board state and uses a heuristic to calculate the value of a board at that moment.

We had to come up with three different heuristics to calculate board states and i used the following rules.

Heuristic 1: Calculates the number of moves that the Active player has. What this does is just maximize the number of moves so that it doesnt get isolated and lose the game.

heuristic 2: This heuristic calculates the difference between the number of moves that the active player has and the number of moves that the inactive player has. What this does is look at how much space the opponent has and tries to minimize their number of moves. in essence it takes the oppositions moves into account. Doing this you could also weight whose moves are more important for a passive or aggressive play.

Heuristic 3: I wanted this calculation to be more dynamic in taking account what part of the game the board state is in. If it's in the beginning, the evaluation is a more passive playstyle, so it tries to maximize the number of moves the active player has and slowly swaps to a move aggressive play style. This is done by seeing how many spaces there are left in the game and if a threshold has been reached it swaps the evaluation function used.

Looking at the tournament results it shows that,from my heuristics, heuristic 2 is the best with the highest winrate 55.7%. This could be that when a certain depth it is still good at predicting the value of the board as if it were at end game. Another reason for why its the best, out of the three is becuase there are no modifiers on how it should so it take both player moves into account equally. The last reason for why heuristic 2 is the best could be becuase its not too simple but not overly complicated. This is becuase heuristic 1 got the lowerst score, meaning that it was too simple. Heuristic 3 is overcomplex since it did slightly worse then 2 even though 3 was based off of 2.

Fig. 1)

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	5	5	9	1	7	3
2	MM_Open	7	3	3	7	6	4	6	4
3	MM_Center	9	1	5	5	9	1	6	4
4	MM_Improved	7	3	2	8	2	8	4	6
5	AB_Open	4	6	3	7	3	7	4	6
6	AB_Center	6	4	1	9	5	5	4	6
7	AB_Improved	4	6	0	10	5	5	2	8
	Win Rate:	64.3%		27.1%		55.7%		47.1%	