For this project I have evaluated **three different heuristic functions**. This document discusses each of them separately and provides a simple summary at the end. Each function was evaluated using the tournament program few times for different configurations of each evaluation function.

The first heuristic function, implemented in **custom_score_3** function, is the simplest of the three. It calculates a difference between the player's and its opponent's number of available legal moves. I call this function a **legal-moves function**. I wanted to start simple by rewarding the player for the number of legal moves available to it, but at the same time penalizing it the higher the opponent's number of legal moves was. This is a very simple strategy with very low compute resource demand. It allows for fast evaluation so there is more time available for depth search, whilst it still provides a rough approximation of the current state of the play. By running various experiments legal-moves function consistently reached around 68% winning rate. It would appear that this function mostly struggled against AB_ opponents which employed similar evaluation functions. I would expect an equal battle with these agents and the experiments proved me right. Given the simplicity of this function, it probably could only be used early on in the game when the game board is open and we don't need to think too much about strategy. I expect this strategy to work way worse at later stage of the game when there are fewer available squares.

The second heuristic function, implemented in **custom_score_2**, is [in theory] slightly more advanced. It came out as a result of my experience of playing isolation in the real life several times against my friends. I noticed that once the knight gets closer to the edge of the board its options to move are limited and often drop very low when it gets into the corner of the game i.e. in between two perpendicular board edges. I decided to model the second evaluation function based on this insight. I call this function the **edge function.** It rewards the positions when the player is away from the board edges whilst at the same time penalises it when the opponent keeps off the edges. too. This in theory could lead to the player pushing the opponents to the edges (i.e. cornering it) whilst at the same time keeping itself away from the edges.
I've used the number of possible edge positions each player has as weights in the final calculation - we can look at this function as a weighted sum where the weights are the available legal edge positions and sum items are all available legal moves. The results seem to show promising direction providing for over 73% average winning rate in 5 tournaments and getting over 70% consistently.

Finally, the last evaluation function, implemented in **custom_score**, is focussing on the distance between the players on the board and the open space. I call this function **distant-space function** This evaluation function rewards the bigger the distance between the two players is as well as the number of open squares on the board. It favours the distance over the blank spaces, by multiplying it with a large constant. The idea behind this evaluation function was to focus more on defence than on the offence like in the edge function mentioned earlier: the player using this function should in theory search for spaces that give it bigger distance from the opponent. This function in its basic form performed worse than the first function which was merely focusing on number of moves. After making a few experiments by multiplying the distance factor with various constants, I've settled on a large number by which I may have forced the agent to favour the distance aggressively over the blank spaces. This had an effect of higher winning rate of around 72% on average in 6 tournaments. This is slightly worse than the second function, but still seems better than the first one.

The results seem to suggest that the edge function provides for the best results, alas more rigorous experiments would have be be done to prove that this is really the case. The worst performance came from the simplest heuristic function, the legal-moves function, which rewards number of legal moves and penalises the same metric of the opponent. Whilst the legal-moves function provides for deeper game tree searches the more sophisticated evaluation functions seem to provide for better results. I also noticed that the more sophisticated evaluation functions on average perform visibly better against the AB_ opponents than the legal-move function, which probably just met its match in AB_ opponents.