# Project Part B Report

## Introduction:

\*\*\* MIGHT NOT NEED TO WRITE THIS

## Agent Design/ Strategy:

Our program seeks to find the best moves in its position by utilising the adversarial game playing algorithm: Minimax with Alpha-Beta pruning. We chose this method as our aim of finding the best move is a classic adversarial search problem where both players compete in a perfect information, static environment. Each player seeks to maximise their own utility function while minimising that of their opponent, so we utilised Minimax to effectively explore all feasible action sequences to a certain depth. To increase the depth that we searched to, we implemented Alpha-Beta pruning which helped avoid exploring branches that we knew lead to sub optimal positions, this was necessary as the number of states grows exponentially with depth, so we sought to eliminate as much expansion as possible to save space and time. This implementation was based on the AIMA textbook; we adapted it to suit our game’s conditions.

Evaluating each position was done by our evaluation function, which heuristically evaluated positions based on the location of player and enemy pieces. The main component of which was our difference in vertical distances sum, this component comprised of the sum of vertical distances for each enemy piece to their respective end of the board (goal state) minus the sum same sum of vertical distances for the player frogs. This provides a simple heuristic that ensures we know which player has the advantage when it comes to proximity to their goal, the heuristic is hence positive when our player s closer to its goal and negative otherwise.

\*\*\* ADD MORE COMPONENTS HERE

When our program was at the stage where it could see forced wins or losses, our heuristic returned infinity or negative infinity respectively. We then adapted our search function to keep track of the depth in which each terminal state was found and propagate it back up the tree. This allowed us to pick the solution at the shallowest depth, so we could win as fast as possible and avoid cycles.

## Algorithmic Enhancements & Constraints:

\*\*\* Talk about space and time complexity, nodes expanded and what we did to make our solution more efficient.

## Performance Evaluation:

\*\*\* Compare results vs Random, Greedy and other engines.

## References: