1. **Member**: Bowei Dong, Ruiyan Song

2. **Important Files and the logic explanation:**

Game.py: It has the game user interface code that would allow the user to see four layers of 4*4 matrix of the 4*4*4 cube. It also detects mouse clicks when the user clicks the screen to make the next move. User can play the game either with a random player or the AI by calling functions in this file. It also further includes the training process of the model and calling updateQ in Qlearning.py to update the utility function. We try to let our model learn through some special conditions as well. If any move is a winning move for any of the player (three in a line already), instead of using utility function, the agent will make the move to either prevent or try to win the game.

Qlearning.py: It has functions that determines the next action of the agent, including random moves or greedy strategy based on the value of utility. It also has a function that updates the utility after each iteration.

The utility function we used is : Q_last + 0.3*((reward + 0.9 * Q_next_max) - Q_last)

Reward: 1 if that player wins, 0.5 if it is a tie, 0 if that player loses

Epsilon: There is a 20% chance that it will explore random actions, and 80% of chance it will make a move that has the highest utility value. If the utility values of certain moves are the same, then we make random moves.

Train.py and **Play.py**: By their name, it creates instances of a game to either train or play. Please follow instruction on github to run.

https://github.com/masteradafa/COMP560HW2/blob/master/README.md

3. **Individual contribution**

Ruiyan Song: writing reports, writing evaluation functions that evaluates whether a certain condition satisfies the ending condition, writing interfaces

Bowei Dong: adding special conditions in training, pair programming together with Ruiyan to develop the entire project, writing the readme file and submitting the project onto github.