OPTION 2: Game Playing Techniques
Team member: Jingqiang Wang, Junhao Zeng

(a) What are the main techniques you are featuring in your project?

The techniques we have done(maybe improve later):

Zobrist Hash:
One of the enhanced techniques used in our project is using the Zobrist hash to hash the state. The idea is creating an 8×8 table, in which each cell is a dictionary, and stored 2 random number. And for the first time we calculate a state value, it will be stored in the dictionary according to the hash value of this state. The next time we meet this state, we will get the state value from the hash dictionary instead of calculating it again. We also try implement another way to hash the state. The only difference is for each cell in the pre-generated table, we store a few random numbers according to the piece placed here.

Complicated static evaluation:
Another enhanced techniques used in our project is using the more complicated static evaluation method to evaluate the board.
First, for a specific status of the board, we count how many enemies all pincer can kill in one move, and adding five points for each kill. Vice versa, we count how many allies can be killed by enemy pincer in one move, and deducting five points for each kill.
Besides, because the more allies a king has in his surrounding, the safer the king is, so we check the surroundings of our king, the surrounding means the eight cells which connect to the king cell. For each surrounding cell, if it is an ally, we add five points, if it is an enemy, we deduct five points. Vice versa, for each surrounding cell of enemy king, if it is an ally, we deduct five points, if it is an enemy, we add five points.
What's more, we count how many allies are frozen by enemy's freezer or imitator, for each frozen ally, we deduct one point. Vice versa, we count how many enemy are frozen by our freezer or imitator, for each frozen enemy, we add one point.
In the end, we add or de a specific points for each piece depending on what it is.

The techniques we are going to implement:
use of machine learning in automatically improving the custom static evaluation function

(b) How far along is your implementation (what works, what does not yet work?).

The Zobrist hash works. However, we need to test the efficiency of both hash approaches. Besides, whether should we use Zobrist hash depends on the complexity of calculating the values of the states. If it is very efficient, then we don't need Zobrist hash. Because we still need time to calculate the hash value.

The complicated static evaluation method can certainly evaluate the board better, it seems can improve the odds of winning, this is what works However, the complicated static evaluation is also time-consuming, it obviously reduces the number of states that we can explore. Therefore, there is a trade off between complexity of static evaluation and the number of explored states and we need to find a balance between them, this is what does not yet work.