We took the driver code from the geneticTSP program we used in class, and adapted it for our new usage in the knight solution driver program. We had to change most of the variable names in order for it to work with the new interface that we designed, and some features such as the find best function were designed to look for the lowest value with the fitness function, which we didn’t realize. All we had to change was the way that it sorted to reverse it, and to change a less than sign to a greater than sign. We had another issue where the interface class was named IndividualGene but one of the groups used the class name Individual and the program was throwing an error. We researched different selection algorithms which could have been promising, but using the selection algorithm from the TSP model gave us good results anyways.

We tested out different settings for the survival ratio and reproduction type ratio with the different groups’ implementations and found for the first group that survival ratio lower (~0.15) and reproduction type ratio higher (~0.9) worked very well. With a few different input coordinates we were able to get the most optimal 64 step path on an 8x8 chess board relatively quickly with these settings.

With group 2’s implementation for some reason no matter what settings we tested, it started at around 25 and climbed slowly up to 30.