A: Problem Statement

The goal of this programming assignment is to implement a simple board game with the data structures we have discussed in class. For this assignment we have to choose the appropriate data structure that delivered the best performance in terms of memory and runtime.

B: Experimental Setup

CPU: Intel Core i5

Clock speed: 2.3 GHz

RAM: 8GB

The machine I used was a 2018 MacBook Pro with 8GB of RAM and a Intel Core i5 CPU with a clock speed of 2.3 GHz and the OS I used was the Mac OS X for testing.

C: Algorithm Design

The data container we decided to use was the stl map class. There seemed to be no reason to use the multimap class because duplicate IDs were not needed. We did not use the stl set class due to the restriction that only primitive data types can be used as values, thus the reason why we used the map class which can support abstract data types.

Run-time complexity of each of the functions in our code:

Insert: N + log(N) = O(N)

Remove: N = O(N)

MoveTo: log(N) + log(N) + N\*log(N) = O(N\*log(N))

PrintByID: N = O(N)

Memory complexity for our implementation as a whole:

Since our stl map implementation does not require empty board position to be stored it allows us to save in most cases the majority of memory that would be used in an array implementation. The amount of memory used would be equal to the number of players and their respective ID keys. 4 bytes (key) + 4 bytes (ID) + 4 bytes (X position) + 4 bytes ( Y position) = 16 bytes (per player). O(16N) = O(N)

D: Test Results

See Results.txt