Chandler Teigen
CPTS 233
PA4 Report

A.

The problem for PA4 was to implement a simple board game that allowed players to move around a board, and remove players from the game by move to the same location as the other player. Because the board size and number of players was allowed to be very large, the storage of the players and their positions needed to be implemented using a data structure that allows for easy lookup of players and positions.

B.

I wrote and tested the code for this assignment on my Macbook Pro 2015 using the VScode IDE and running the code on the UNIX terminal on my Macbook. My computer is running MacOS Mojave version 10.14.2, has a 2.7GHz Intel i5 processor, and 8GB of RAM. The code was also tested on the EECS Linux servers, using the newest devtoolset.

C.

Design Document:

Data Containers Used:

Std::set

The std::set container was used to store the Player objects in the game in two different manners. The first std::set called stored the Player objects in order of their IDs and the other stored the same Player objects in order of their positions on the board. I chose the std::set because it is usually implemented as a red-black tree, which has log(n) time-complexity for all of its operations other than printing. This seemed advantageous for the searching, removal and insertion that would be required to run the game. The two different orderings were chosen so that the players could be searched for in log(n) time by both the ID and the position. Even though each std::set had to be searched separately, searching or inserting into both would still be significantly faster than having to search an std::set ordered by ID while looking for a certain location, or vice versa.

Run-time complexities of functions:

Board Member functions:

Function	Run-time complexity O
is_in_bounds	k

insert_player	log(n)
remove_player	log(n)
find_player	log(n)
move_player	log(n)
print_by_ID	n

All accessors, mutators,constructors, destructors for the Board class have O(1) time-complexity.

All functions for the Player class, and Vec2 class have O(1) time-complexity. The memory complexity of my design is constant with respect to m, the board dimension, and linear with respect to n, the number of players.