REPORT

CptS 223 Project #3

Xiyu Xie

A: Problem statement.

Implementing a board game, practice with STL data structures for balanced binary search trees.

B: Experimental setup.

Windows, Microsoft Visual Studio 2008

C:Algorithm design

Class Board:

1. **container: Memory Complexity O(N)**

map <ID, player>	Players	O(N)	// used by call functions
map< "(X,Y)", ID>	Locations	O(N)	// make location checking easier
			// for function insert and moveto

2. **function Time Complexity**

Insert (ID, X, Y) O(logN)

Check is board full, out of board and others	O(1)	
Players.find (ID)	O(logN)	// check ID conflict
Locations.find(" (X,Y) ")	O(logN)	// check location conflict
if (all satisfied)		
Players.insert(<ID, player>)	O(logN)	
Locations.insert(<"(X,Y)" , ID>)	O(logN)	
N++		
return true		
return false		

Remove(ID) O(logN)

iter=Players.find (ID)	O(logN)	// check exist
if (iter!= Players.end)		
location<- iter.player.location	O(1)	// get location
Players.erase (iter)	O(1)	// remove from Players
Locations.erase(location)	O(logN)	// remove from Locations
N--;		
return true		
return false		

MoveTo (ID, X2, Y2) O(logN)

within board check	O(1)	
iter=Players.find (ID)	O(logN)	// check exist

```
if (iter!= Players.end )
        location<- iter.player.location          O(1)          // get location
        check is movement legal                  O(1)
        if (all satisfied)
                iter2=Locations.find("(X2,Y2)")  O(logN)       // check if occuptied
                ID2=iter2.ID;
                Remove(ID2);                     O(logN)
                Remove(ID)                       O(logN)
                Insert(ID)                       O(logN)
                return true
        return false
```

PrintByID: **O(N)**

in-order traversal of Players