Name: Sungmin Cho

UCID: 205124669

Prof. Ambrosio

1. Description: I chose to do a simple doubly linked list to code the program for BballRoster class because it gives more variety in terms of possible code designs and improves the efficiency of the program. My list neither is circular nor have dummy nodes. My node contains Node pointer next and previous and string variable firstName and lastName and SomeType variable value.
2. Obstacles: My biggest struggle was figuring out the design of signPlayer function. At first, I finished everything without thinking about the exceptional cases. Soon, I came to realize that I missed the exceptional case of signing a player to tail among several others in the list with same last name.
3. Pseudocode:

bool signPlayer(const string& firstName, const string& lastName, const SomeType& value)

if the roster is empty

dynamically allocate a new node and set its full name and value as the parameter input

set its previous and next node pointer as null pointer

set it as head and tail

return true

return false if there’s a player with the same full name.

otherwise

dynamically allocate a new node and set its full name and value as the parameter input

looping through every node

when last name with a lower alphabetical order found

place the new node in front of the node with that last name

set the new node as head if necessary

return true

if there is a same last name

loop from the names with same last name

when first name with lower alphabetical order found

place the new node in front of the node with that first   
name

set the new node as head if necessary

return true

when the new node must be placed at the very back of the list

place it at the very back and set it as tail

return true

place the new node after the last node with the same last name

return true

if the new node is the last in alphabetical order

place it at the very back, set it as tail

return true

bool renouncePlayer(const string& firstName, const string& lastName)

return false if there’s no player with the input full name

otherwise

looping through every node in the list

when a node with same full name detected, delete it and return true

bool playerOnRoster(const string& firstName, const string& lastName) const

return false if the list is empty

otherwise

loop through every node

return true if there is a node with same full name as the input

return false

bool joinRosters(const BballRoster& bbOne, const BballRoster& bbTwo, BballRoster& bbJoined)

if bbOne is empty, set bbJoined as bbTwo and return true

else if bbTwo is empty, set bbJoined as bbOne and return true

otherwise

set bbJoined as the bigger between bbOne and bbTwo and BballRoster small as the   
smaller

looping through the nodes in small

find the node’s first name, last name, and the value

lookup in bbJoined to see if there’s any player with the same full name and   
 check if the two nodes’ values are the same

renounce the player and set returnVal as false

otherwise sign the player from small into bbJoined

return returnVal(which would be true unless changed in the above line)

1. Test cases:

BballRoster mwp;

assert(mwp.signPlayer("Adam", "Wright", 41)); // test initial signing

assert(mwp.signPlayer("Jason", "Falitz", 37)); // test nonempty list signing

assert(mwp.howManyPlayers() == 2);

string first, last;

int ac;

assert(mwp.choosePlayer(0, first, last, ac) && ac == 37); // test choosePlayer

assert(mwp.choosePlayer(1, first, last, ac) && (first ==

"Adam"

&& ac == 41));

BballRoster a, b, ab;

a.signPlayer("Kyrie", "Irving", 11);

a.signPlayer("Kevin", "Durant", 7);

a.signPlayer("DeAndre", "Jordan", 6);

b.signPlayer("Kevin", "Durant", 7);

b.signPlayer("Spencer", "Dinwiddie", 8);

joinRosters(a, b, ab);

assert(ab.howManyPlayers()==4); // test joinRoster 2 nodes with same values

b.resignPlayer("Kevin", "Durant", 35);

joinRosters(a, b, ab);

assert(ab.howManyPlayers() == 3); // test joinRoster 2 nodes with same name but

different value

BballRoster clippers, result;

clippers.signPlayer("Brook", "Lopez", 11);

clippers.signPlayer("Robin", "Lopez", 8);

clippers.signPlayer("Greek", "Freak", 34);

string f, l;

SomeType v;

assert(clippers.choosePlayer(2, f, l, v) && f == "Robin" && l == "Lopez");

// test signing with first name alphabetical

checkRoster("\*", "Lopez", clippers, result);

assert(result.howManyPlayers() == 2); // test checkRoster with wildcard first name  
 checkRoster("\*", "\*", clippers, result);

assert(result.howManyPlayers() == 3); // test checkRoster with both wildcards  
 checkRoster("Greek", "\*", clippers, result);

assert(result.howManyPlayers() == 1); // test checkRoster with last name wildcard