==================================

Name: Nevin Liang

UCLA ID: 705-575-353

Subject: Project 2 Writeup

Date: July 6, 2020

==================================

Design of Implementation:

----------------------------------------

* List takes the form of a standard linked list with Node pointers connected with “next” and “prev” pointers.
* I chose this form because it is the easiest to implement especially with removal and insertion
* No tail pointer because none of the function implementations I coded required one
  + One method, addInvitee, required the tail pointer, but I stopped the while loop when ->next of the current iterator was `nullptr`
* 
* (I apologize for the bad drawing :P)

Notable Obstacles:

------------------

* Figuring out how to begin implementing the linkedlist was definitely a difficult task
* Figuring the two non-member functions was probably the hardest part of the program algorithmically
* Making sure there were no memory leaks was also difficult. I decided to use the copy-and-swap idiom for the assignment operator because that would help me with multiple areas.
  + First of all, it would make sure the destructor was able to take care when using the assignment op
  + Also, I was able to kill two birds with one stone when I implemented the two non-member functions
    - I had to prevent a memory leak or malfunction when the result and one of the other parameters pointed to the same value.
    - I used a temp variable to calculate the answer, and then copied it into the result variable. If the result variable already held something, one had to delete it, but my assignment operator took care of that.

Pseudocode for Non-trivial Algorithms:

--------------------------------------

* addInvitee
  + Create new node with parametric values
  + Check if head = nullptr
    - If yes -> set head as new node
  + Else
    - Loop through entire list
      * Check if last name < current last name
      * Or if last name = current last name
        + If yes -> insert node

Two cases: head node and not head node

Head: adjust iter’s prev and n’s ptrs

Not: adjust iter’s prev & next and n’s

* + - * Check if full name matches
        + If yes -> return false
      * Check if next is nullptr because possible insertion at end
        + If yes -> set temp tail ptr to current iter
  + If reached here, must add to end so use tail ptr
    - Adjust tail ptr and connect n’s ptrs.
* addOrModify
  + Call addInvitee
    - If returns true
      * Just return true it has been added
    - If returns false
      * Return true but also call modifyInvitee to change its value
* dropFromGuestList aka remove
  + Loop through entire list
    - If pointer’s values match
    - 3 cases
      * Current iter = head
        + Adjust head
        + Delete iter
        + (if head is not nullptr)

Set head’s prev ptr to nullptr

* + - * Current iter = tail
        + Set iter’s prev’s next ptr to nullptr
        + Delete iter
      * Current iter = middle
        + Adjust the neighboring nodes’ pointers
        + Delete iter
* combineGuestLists
  + Create temporary linkedlist temp
  + Set temp to bpOne
  + Loop through bpTwo
    - Check if person in bpTwo exists in temp
      * If yes -> check values match
        + If yes -> flag is false
      * If no call addInvitee to add to linkedlist
  + Set bpOne to result
    - Prevent aliasing errors
* verifyGuestList
  + (Similar to previous function)
  + First create temporary linkedlist temp
  + Loop through bpOne
    - Check if current person matches parametric values
      * (fsearch == first || fsearch == "\*") && (lsearch == last || lsearch == "\*")
    - If yes, add to temp using addInvitee
  + Set bpResult to temporary linkedlist to prevent alias errors

Test Cases Used for Thorough Testing

------------------------------------

T1: basic test to test addInvitee and selectInvitee and whosOnTheGuestList

T2: Testing dropFromGuestList function

T3: testing combineList function with normal inputs

T4: testing combineList with conflicting info for 2 bp’sm

T5: testing verifyGuestList with 1 wildcard

T6: testing verifyGuestList with 1 other wildcard, 2 wildcards, and 2 nonexistent names

T7: testing verifyGuestList with result and original BP as the same BP

T8: testing combinedGuestList with result and 1 of the original BP’s as the same BP

T9: testing removing head pointer and empty list

**void** t1() {

BirthdayParty theLastDance;

theLastDance.addInvitee ("Michael", "Jordan", 23);

theLastDance.addInvitee ("Scottie", "Pippen", 33);

theLastDance.addInvitee ("Dennis", "Rodman", 91);

theLastDance.addInvitee ("Luc", "Longley", 13);

theLastDance.addInvitee ("Ron", "Harper", 9);

**for** (**int** n = 0; n < theLastDance.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

theLastDance.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t2() {

BirthdayParty dodgers;

dodgers.addInvitee("Clayton", "Kershaw", 31.0);

dodgers.addInvitee("Cody", "Bellinger", 11.5);

assert(!dodgers.personOnGuestList ("",""));

dodgers.addInvitee("Mookie", "Betts", 27.0);

dodgers.addInvitee("", "", 0.57);

dodgers.addInvitee("Justin", "Turner", 20.0);

assert(dodgers.personOnGuestList ("", ""));

dodgers.dropFromGuestList("Mookie", "Betts");

assert(dodgers.whosOnTheGuestList() == 4 && dodgers.personOnGuestList("Clayton", "Kershaw")

&& dodgers.personOnGuestList ("Cody", "Bellinger") && dodgers.personOnGuestList ("Justin", "Turner")

&& dodgers.personOnGuestList ("", ""));

}

**void** t3() {

BirthdayParty bpOne;

bpOne.addInvitee("Kobe", "Bryant", 8);

bpOne.addInvitee("AC", "Green", 45);

bpOne.addInvitee("Shaquille", "Oneal", 34);

BirthdayParty bpTwo;

bpTwo.addInvitee("Kobe", "Bryant", 8);

bpTwo.addInvitee("Horace", "Grant", 54);

BirthdayParty bpJoined;

cout << combineGuestLists(bpOne, bpTwo, bpJoined) << endl;

**for** (**int** n = 0; n < bpJoined.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

bpJoined.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t4() {

BirthdayParty bpOne;

bpOne.addInvitee("Kobe", "Bryant", 8);

bpOne.addInvitee("AC", "Green", 45);

bpOne.addInvitee("Shaquille", "Oneal", 34);

BirthdayParty bpTwo;

bpTwo.addInvitee("Kobe", "Bryant", 24);

bpOne.addInvitee("Pau", "Gasol", 16);

BirthdayParty bpJoined;

cout << combineGuestLists(bpOne, bpTwo, bpJoined) << endl;

**for** (**int** n = 0; n < bpJoined.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

bpJoined.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t5() {

BirthdayParty memories;

memories.addInvitee("Gianna", "Bryant", 13);

memories.addInvitee("Kobe", "Bryant", 41);

memories.addInvitee("Little", "Richard", 87);

memories.addInvitee("Jerry", "Stiller", 92);

BirthdayParty bpResult;

verifyGuestList("\*", "Bryant", memories, bpResult);

**for** (**int** n = 0; n < bpResult.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

bpResult.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t6() {

BirthdayParty moreMemories;

moreMemories.addInvitee("Kirk", "Douglas", 103);

moreMemories.addInvitee("Fred", "Neal", 77);

moreMemories.addInvitee("Pop", "Smoke", 20);

moreMemories.addInvitee("Fred", "Willard", 86);

BirthdayParty result;

verifyGuestList("Fred", "\*", moreMemories, result);

**for** (**int** n = 0; n < result.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

result.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

verifyGuestList("\*", "\*", moreMemories, result);

**for** (**int** n = 0; n < result.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

result.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

BirthdayParty result2;

result2.addInvitee("Bob", "Joe", 103);

verifyGuestList("ASDF", "ASDF", moreMemories, result2);

**for** (**int** n = 0; n < result2.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

result2.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t7() {

BirthdayParty moreMemories;

moreMemories.addInvitee("Kirk", "Douglas", 103);

moreMemories.addInvitee("Fred", "Neal", 77);

moreMemories.addInvitee("Pop", "Smoke", 20);

moreMemories.addInvitee("Fred", "Willard", 86);

BirthdayParty result;

verifyGuestList("Fred", "\*", moreMemories, moreMemories);

**for** (**int** n = 0; n < moreMemories.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

moreMemories.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t8() {

BirthdayParty bpOne;

bpOne.addInvitee("Kobe", "Bryant", 8);

bpOne.addInvitee("AC", "Green", 45);

bpOne.addInvitee("Shaquille", "Oneal", 34);

BirthdayParty bpTwo;

bpTwo.addInvitee("Kobe", "Bryant", 24);

bpOne.addInvitee("Pau", "Gasol", 16);

BirthdayParty bpJoined;

cout << combineGuestLists(bpOne, bpTwo, bpTwo) << endl;

**for** (**int** n = 0; n < bpTwo.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

bpTwo.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}

**void** t9() {

BirthdayParty bpOne;

bpOne.addInvitee("Kobe", "Bryant", 8);

bpOne.addInvitee("AC", "Green", 45);

bpOne.addInvitee("Shaquille", "Oneal", 34);

bpOne.modifyInvitee("AC", "Green", 100);

bpOne.dropFromGuestList("Shaquille", "Oneal");

bpOne.dropFromGuestList("Kobe", "Bryant");

**int** test;

cout << bpOne.noInvitees() << endl;

cout << bpOne.checkGuestList("Kobe", "Bryant", test) << endl;;

**for** (**int** n = 0; n < bpOne.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

bpOne.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

cout << bpOne.dropFromGuestList("AC", "Green") << endl;

**for** (**int** n = 0; n < bpOne.whosOnTheGuestList(); n++) {

string first;

string last;

**int** val;

bpOne.selectInvitee (n, first, last, val);

cout << first << " " << last << " " << val << endl;

}

}