File: EE599_lab8_Yao_Guo.zip
USCID: 2176023892
Email: yaog@usc.edu
Description:

Task 1:
Add a final two tests to testReadFrom() method that reads the fourth and fifth Members from our input file.

```
// read fourth Member in input file
getline(fin, separator);
member.readFrom(fin);
assert( member.getUsername() == "newmembers" );
assert( member.getName() == "Mark Brown" );
assert( member.getDateBirth() == "03/20/1990" );
assert( member.getEmail() == "mark320@gmail.com" );
assert( member.getPhoneNumber() == "2133900110"  );
assert( member.getYear() == 2015);
cout << " 3 " << flush;

// read fifth Member in input file
getline(fin, separator);
member.readFrom(fin);
assert( member.getUsername() == "davidclubname" );
assert( member.getName() == "David Smith" );
assert( member.getDateBirth() == "01/10/1989" );
assert( member.getEmail() == "davidsmith@gmail.com" );
assert( member.getPhoneNumber() == "2551020510"  );
assert( member.getYear() == 2018);
cout << " 4 " << flush;
```

Task 2:
Using this TDD concept and this working model that we have designed to implement three Memberlist’s operations, **implement search by username, search by email and search by registered year.**

First, write the test code prototype:

```
void MemberListTester::testSearchByUsername(){
    cout <<"- Testing search by username..." <<flush;

    // cout<< " 0 " <<flush;

    cout<< " Passed!" << endl;
}

void MemberListTester::testSearchByEmail(){
    cout <<"- Testing search by Email..." <<flush;

    // cout<< " 0 " <<flush;

    cout<< " Passed!" << endl;
}

void MemberListTester::testSearchByYear(){
    cout <<"- Testing search by Registered Year..." <<flush;

    // cout<< " 0 " <<flush;

    cout<< " Passed!" << endl;
}
```

Also, in MemberList.cpp and MemberList.h, we add the corresponding function.

```
    Member* searchByUsername(string name);
    Member* searchByEmail(string email);
    vector<Member>* searchByYear(unsigned year);
```

Then, we define our test code;

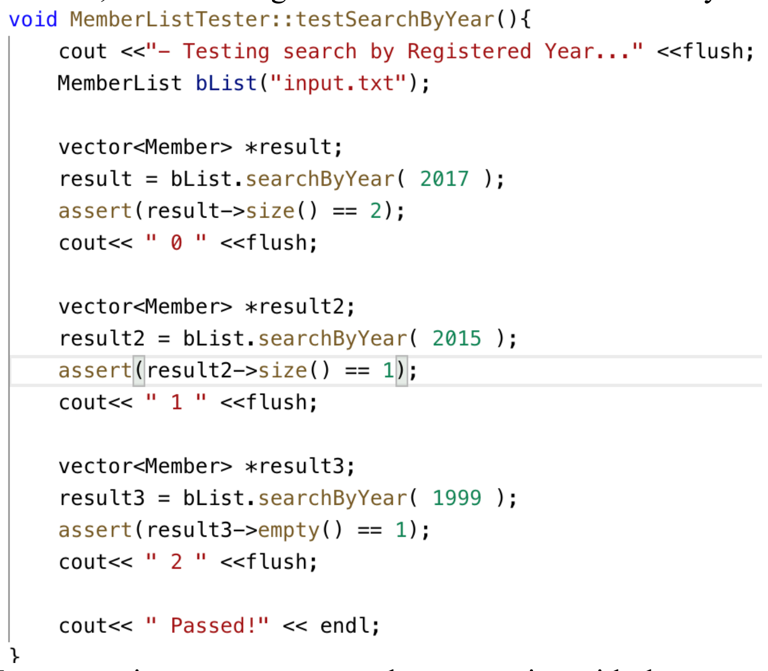
```
void MemberListTester::testSearchByEmail(){
    cout <<"- Testing search by Email..." <<flush;
    MemberList bList("input.txt");

    Member *member = bList.searchByEmail("janescott@gmail.com");
    assert( member->getUsername() == "Spiderman" );
    assert( member->getName() == "Jane Scott" );
    assert( member->getDateBirth() == "02/17/1998" );
    assert( member->getEmail() == "janescott@gmail.com" );
    assert( member->getPhoneNumber() == "2133770909"  );
    assert( member->getYear() == 2017);
    cout<< " 0 " <<flush;
```

We need the searchByEmail to return a pointer of a member object, if we didn’t find the name, we return NULL.
Same as searchByUsername;

```
Member* MemberList::searchByEmail(string email){
    unsigned num = getNumMembers();
    int i;
    for(i=0; i<num; i++){
        if(totalMembers[i].getEmail() == email){
            return &totalMembers[i];
        }
    }
    return NULL;
}
```

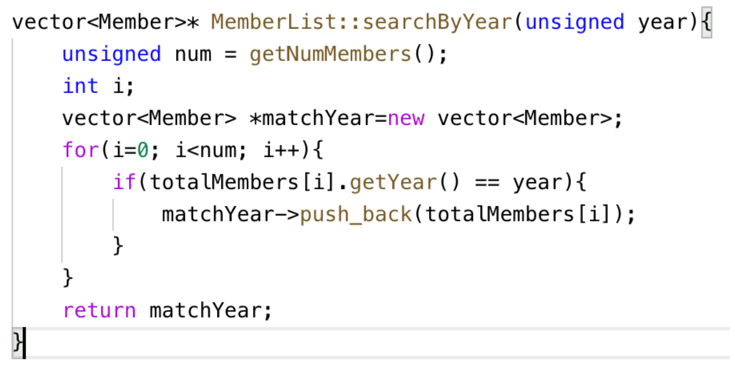
In this search function, if we find the matched object, we return the corresponding member object via pointer. We assume that everyone's email and username is unique so that we only need to return the pointer of that member object.

However, test searchByYear is different. Because the year is not unique, so we need a container to save the matched year member objects, then we return a C++ reference.
To do so, the following is the test function and searchByYear function.



```
void MemberListTester::testSearchByYear(){
    cout <<"- Testing search by Registered Year..." <<flush;
    MemberList bList("input.txt");

    vector<Member> *result;
    result = bList.searchByYear( 2017 );
    assert(result->size() == 2);
    cout<< " 0 " <<flush;

    vector<Member> *result2;
    result2 = bList.searchByYear( 2015 );
    assert(result2->size() == 1);
    cout<< " 1 " <<flush;

    vector<Member> *result3;
    result3 = bList.searchByYear( 1999 );
    assert(result3->empty() == 1);
    cout<< " 2 " <<flush;

    cout<< " Passed!" << endl;
}
```

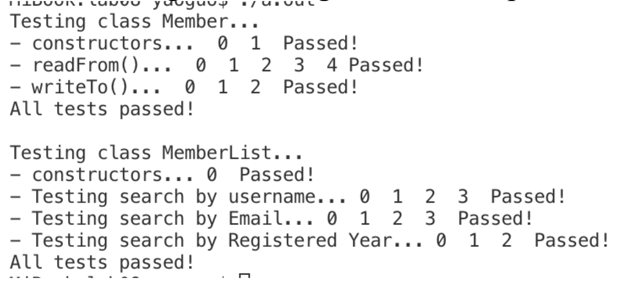
For convenience, we compare the vector size with the correct matched size.



```
vector<Member>* MemberList::searchByYear(unsigned year){
    unsigned num = getNumMembers();
    int i;
    vector<Member> *matchYear=new vector<Member>;
    for(i=0; i<num; i++){
        if(totalMembers[i].getYear() == year){
            matchYear->push_back(totalMembers[i]);
        }
    }
    return matchYear;
}
```

If the vector is empty, which indicate we didn't find any matched year member objects.

After finishing these, we get our test output:



```
Testing class Member...
- constructors...  0  1  Passed!
- readFrom()...  0  1  2  3  4 Passed!
- writeTo()...  0  1  2  Passed!
All tests passed!

Testing class MemberList...
- constructors... 0  Passed!
- Testing search by username... 0  1  2  3  Passed!
- Testing search by Email... 0  1  2  3  Passed!
- Testing search by Registered Year... 0  1  2  Passed!
All tests passed!
```

Reference:

This Lab8’s instruction.