Project 4 Report

I believe that I was able to finish all parts of the project according to the specifications with the correct STL components without any bugs.

Class PersonProfile

* I made a PersonProfile object and inserted the name Annie and the email Annie.net, then called the GetName() and GetEmail() to make sure both functions were behaving properly
* I then created AttValPairs, and added that to the Person through the AddAttValPair function
* Then I called the GetNumAttValPairs() function to see if the number of AttVal pairs that I added amounted correctly
* Lastly I checked by GetAttVal function by printing out it’s contents to see if the behavior was correctly implemented

PersonProfile x("annie", "gmail");

std::cout << x.GetName() << std::endl;

std::cout << x.GetEmail() << std::endl;

AttValPair a1("hobby", "hiking");

AttValPair a2("hobby", "walking");

AttValPair a3("hobby", "jogging");

AttValPair a4("hobby", "hiking");

x.AddAttValPair(a1);

x.AddAttValPair(a2);

x.AddAttValPair(a3);

x.AddAttValPair(a4);

std::cout << x.GetNumAttValPairs() << std::endl;

for(int k=0; k!=x.GetNumAttValPairs(); k++){

AttValPair a;

x.GetAttVal(k, a);

std::cout<< a.attribute<< "->" << a.value << std::endl;

}

Class MemberDatabase

* I made my own versions of members.txt to test the LoadDatabase function, for example I made a version with only 5 people and checked if that loaded correctly
* Then I created a version with duplicate members to test if that would load with the correct behavior or not
* To check if this was working I had a print statement in my main function
* I then check to see if my GetMemberByEmail function was working properly by testing if it would print the correct corresponding name
* I then checked my FindMatchingMembers function by checking the size of the vector returned

MemberDatabase md;

if(md.LoadDatabase("/Users/nidhisunkam/Downloads/Unhinged/members.txt")){

std::cout << "can load" << std::endl;

}

else{

std::cout << "cant load" << std::endl;

}

std::cout << (\*at1.GetMemberByEmail("CoFish1380@msn.com")).GetName() << std::endl;

std::cout << (\*at1.GetMemberByEmail("AlvinHin5@yandex.com")).GetName() << std::endl;

std::cout << (\*at1.GetMemberByEmail("Annie@g.ucla.edu")).GetName() << std::endl; //this is not an actual email

Class AttributeTranslator

​​ - To check if this was working I had a print statement in my main function

* I then checked my FindCompatibleAttValPairs function by checking the size of the vector returned

AttributeTranslator at;

if(at.LoadDatabase("/Users/nidhisunkam/Downloads/Unhinged/translator.txt")){

std::cout << "can load" << std::endl;

}

else{

std::cout << "cant load" << std::endl;

}

Class MatchMaker

* I tested the IdentifyRankedMatches function by putting in various email addresses of people in the database at different thresholds and then comparing it to the sample output given by the curl command in the spec
* I inserted [AbFow2483@charter.net](mailto:AbFow2483@charter.net), with a threshold of 7 then checked with the sample project to check for correctness
* I inserted [AbFow2483@charter.net](mailto:AbFow2483@charter.net), with a threshold of 6 then checked with the sample project to check for correctness
* I inserted [AbFow2483@charter.net](mailto:AbFow2483@charter.net), with a threshold of 4 then checked with the sample project to check for correctness
* I inserted [AbFow2483@charter.net](mailto:AbFow2483@charter.net), with a threshold of 1 then checked with the sample project to check for correctness
* I inserted MHa7@sky.com, with a threshold of 1 then checked with the sample project to check for correctness
* I inserted MHa7@sky.com, with a threshold of 7 then checked with the sample project to check for correctness
* (I continued this process for multiple emails, and various thresholds)

Class RadixTree

* I repeatedly inserted items into the RadixTree and searched for them to check if the tree was behaving properly
* I also made a print function to test if the tree was holding the right prefixes at each level

RadixTree <std::set<std::string>> rad;

RadixTree <std::string> rad;

std::set<std::string> one;

one.insert("dancing");

one.insert("writing");

one.insert("sports");

rad.insert("hobby", one);

std::set<std::string> two;

std::string two;

two.insert("pizza");

two.insert("cookie");

two.insert("watermelon");

rad.insert("favorite food", two);

RadixTree <std::string> rad;

rad.insert("tank", "ab");

rad.insert("test", "aa");

rad.insert("tester", "bb");

rad.insert("angry", "cd");

rad.insert("anvil", "de");

rad.insert("ant", "ef");

RadixTree<std::string> a;

a.insert("hobby", "climbing");

a.insert("hobby", "walking");

a.insert("height", "tall");

a.insert("tes", "mouse");

a.insert("test", "howdy");

std::cout << (\*a.search("hobby")) << std::endl;

std::cout << (\*a.search("height"))<< std::endl;

std::cout << (\*a.search("test"))<< std::endl;

std::cout << (\*a.search("tes"))<< std::endl;