David Lin

205770795

Project 4 Unhinged – Report

**PersonProfile Test Cases**

-Ensures that the PersonProfile object is constructed properly and that GetName() and GetEmail() work

-Ensures that AddAttValPairs() works and that non-duplicate AttValPairs are added.

-Ensures that GetNumAttValPairs() and GetAttVal() works

PersonProfile y("Dave", "davertito@gmail.com");

assert(y.GetName() == "Dave");

assert(y.GetEmail() == "davertito@gmail.com");

AttValPair a("hobby", "dancing");

AttValPair b("hobby", "sitting");

AttValPair c("hobby", "basketweaving");

AttValPair g("trait", "evil");

AttValPair e("occupation", "fisherman");

AttValPair f("occupation", "janitor");

AttValPair j("trait", "salacious");

y.AddAttValPair(a);

y.AddAttValPair(b);

y.AddAttValPair(c);

y.AddAttValPair(g);

y.AddAttValPair(e);

y.AddAttValPair(f);

y.AddAttValPair(j);

assert(y.GetNumAttValPairs() == 7);

y.AddAttValPair(a);

assert(y.GetNumAttValPairs() == 7);

AttValPair d("jobbr", "yobb");

y.GetAttVal(0, d);

assert(d.attribute == "hobby" && d.value == "dancing");

y.GetAttVal(1, d);

assert(d.attribute == "hobby" && d.value == "sitting");

y.GetAttVal(2, d);

assert(d.attribute == "hobby" && d.value == "basketweaving");

y.GetAttVal(3, d);

assert(d.attribute == "trait" && d.value == "evil");

y.GetAttVal(4, d);

assert(d.attribute == "occupation" && d.value == "fisherman");

y.GetAttVal(5, d);

assert(d.attribute == "occupation" && d.value == "janitor");

y.GetAttVal(6, d);

assert(d.attribute == "trait" && d.value == "salacious");

assert(y.GetNumAttValPairs() == 7);

**AttributeTranslator Test Cases**

-Ensures that the Load() function works

AttributeTranslator at;

const string afile = "translator.txt";

assert(at.Load("transl.txt") == false);

-Ensures that FindCompatibleAttValPairs() works

AttValPair source("trait", "obnoxious");

std::map<string, AttValPair> attPair;

AttValPair bowling("hobby", "bowling");

AttValPair historian("job", "historian");

attPair.insert({ "hobby", bowling });

attPair.insert({ "job", historian });

vector<AttValPair> vl = at.FindCompatibleAttValPairs(source);

assert(vl.size() == 2);

for (int i = 0; i < vl.size(); i++) {

assert(attPair.find(vl[i].attribute)->second.value == "bowling" || attPair.find(vl[i].attribute)->second.value == "historian");

}

**MemberDatabase Class Test Cases**

-Ensures that the LoadDatabase() function works

MemberDatabase mb;

const string mfile = "members.txt";

assert(mb.LoadDatabase(mfile) && !mb.LoadDatabase("membs.txt"));

In mems.txt, I put two profiles, [muk@gmail.com](mailto:muk@gmail.com) and [jumpa@gmail.com](mailto:jumpa@gmail.com) with the librarian job. mems.txt contains the following profiles.

Abdullah Fowler

AbFow2483@charter.net

3

job,architect

trait,dumb

hobby,knitting

Steeve Giang

sGv@gmail.com

2

job,janitor

trait,dumb

Jun Taka

jumpa@gmail.com

2

job,librarian

trait,evil

Muk Go

muk@gmail.com

2

job,librarian

trait,dumb

Loosey Goosey

losegoes@gmail.com

3

job,chef

trait,nice

hobby,dancing

- Ensures that the FindMatchingMembers() function works

MemberDatabase kb;

const string pfile = "mems.txt";

kb.LoadDatabase(pfile);

AttValPair mince("job", "librarian");

vector<string> vect = kb.FindMatchingMembers(mince);

assert(vect.size() == 2);

for (int i = 0; i < vect.size(); i++) {

cout << vect[i] << endl;

}

- Ensures that the GetMemberByEmail() function works

assert(kb.GetMemberByEmail(“muk@gmail.com”)->GetName() == “Muk Go”);

assert(kb.GetMemberByEmail(“jumpa@gmail.com”)->GetName() == “Jun Taka”);

assert(kb.GetMemberByEmail(“losegoes@gmail.com”)->GetName() == “Loosey Goosey”);

**MatchMaker Class Test Cases**

-I made two attribute value pairs with one that was compatible with itself and the other pair. I gave both pairs to one person and the compatible pair one another person. When using IdentifyRankedMatches(), my MatchMaker class only found 1 match between the two people.

-I looked at [AbFow2483@charter.net](mailto:AbFow2483@charter.net) and [ABran1@sky.com’s](mailto:ABran1@sky.com’s) compatible attributes and found that their matches were the same as what my program found.

-I made one person in the database compatible with everyone else and found 99999 matches. This was to make sure that a person couldn’t match with themselves and that the emails were ordered correctly.

**RadixTree Class Test Cases**

-I added water, then searched for it.

-I added slow, then searched for slow and water to make sure both were in the Radix Tree.

-I added **slow**er, then searched for slow and slower to make sure both were in the Radix Tree.

-I added waster, then searched for water and waste to make sure both were in the Radix Tree.

-I added watch, then searched for water, waste, and watch to make sure all were in the Radix Tree.

-I searched for wa, and t to make sure both were not in the Radix Tree.

-I added wa, then searched to make sure it was in the Radix Tree.