**Project 4 Report**

* **What was I unable to complete?**

I was able to complete all the required implementations in the project

* **Used banned STL components**

None of the files uses a banned STL

* **Bugs I haven’t been able to fix**

When running on Visual C++, it gives a bug of .pdb files not loaded. After interacting with Professor Smallberg, I found out that this is a Microsoft artifact issue and even after following his steps to avoid the issue, I was not able to run the code.

Fun fact: Code runs perfectly for a SMALL members.txt but gives issue for a large members file. There could be a chance my loading memberdatabase is not happening properly using the Radix (It was working with my Fake RadixTree)

**Testing**

* Radix Tree

I checked the Radix tree by inserting similar words starting with slow then slower then slo and slope. This ensured that my Radix tree is successfully creating different branched for each word according to the spec. I also used the search function for these elements to check the correct assigning of the ValueType to each key.

RadixTree<int> rt;

rt.insert("slower", 23);

rt.insert("slow", 45);

rt.insert("slo", 77);

int\* p = rt.search("slow");

std::cout << \*p;

RadixTree<AttValPair> rt;

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

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

rt.insert(a1.attribute, a1);

rt.insert(a2.attribute, a2);

AttValPair\* p = rt.search("hobby");

std::cout << p->value;

* Person Profile

I manually created a new Person to try insetting new attval pairs and check if its successfully inserted in that person.

* MemberDatabase

The first test case was simply loading the memberdatabase file. Next was looking for a specific email using getMemberByEmail in the database example : BrAlliso14214@yandex.com . Then I extracted this person’s attval pairs using the GetAttVal function and keeping them in a locally created vector to confirm the correct addition of the attval pairs.

Another test to check the FindMatchingMembers function was using a common attval pair like job, architect and search for all the members which have that pair. This checks the entire MemberDatabase class

/\*MemberDatabase mdb;

std::cout << mdb.LoadDatabase(MEMBERS\_FILE);\*/

const PersonProfile\* tempman = mdb.GetMemberByEmail("BrAlliso14214@yandex.com");

std::vector<AttValPair> compatiblepairs;

for (int i = 0; i < tempman->GetNumAttValPairs(); i++) {

AttValPair atv;

tempman->GetAttVal(i, atv);

compatiblepairs.push\_back(atv);

} //next print compatible pairs

* Attribute Translator

Like database class, I first loaded this file which was successful in the first try. Then I checked for all the compatible attval pairs to job, architect to ensure correct reading of the translator.txt. I stored all the pairs in a vector (since that is what this function returns) and finally checked the content of this vector.

AttributeTranslator ab;

ab.Load(TRANSLATOR\_FILE);

std::vector<AttValPair> test;

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

test = ab.FindCompatibleAttValPairs(source);

for (int i = 0; i != test.size(); i++)

{

std::cerr << test[i].attribute << std::endl;

std::cerr << test[i].value << std::endl;

}

* Match Maker

To test this function the only viable way was to test first entire code using sample member text file and then with the actual member text file. Proper checking of this class required me to execute the project using the main.cpp provided in the project. So I checked for a person and cross checked with every person who was matching with my input.