**Broken/Incorrect Parts of the Program**

As far as I am aware, my program is finished, uses only allowed STL containers, and has no bugs (but there are probably some I am unaware of)

**Test Cases:**

**RadixTree:**

RadixTree<int> tester;

tester.insert("Jackson", 1);

// Testing that the correct value is found

assert(\*tester.search("Jackson") == 1);

// Testing that keys with similar names are not found

assert(tester.search("Jackso") == nullptr);

// Testing that keys with totally different names are not found

assert(tester.search("Gurbir") == nullptr);

// Adds a key with a very similar name and makes sure values can still be found

tester.insert("Jackso", 2);

assert(\*tester.search("Jackson") == 1);

assert(\*tester.search("Jackso") == 2);

assert(tester.search("Jack") == nullptr);

// Making sure it works with the same start but even shorter

tester.insert("Jack", 3);

assert(\*tester.search("Jackson") == 1);

assert(\*tester.search("Jackso") == 2);

assert(\*tester.search("Jack") == 3);

// Adds a whole new name

tester.insert("Gurbir", 4);

assert(\*tester.search("Gurbir") == 4);

// Makes sure it works with spaces and longer versions of existing keys

tester.insert("Gurbir Randhawa", 5);

assert(\*tester.search("Gurbir") == 4);

assert(\*tester.search("Gurbir Randhawa") == 5);

// Makes sure if the start is the same but the end is different values are able to be found

tester.insert("Gurbalicious", 6);

assert(\*tester.search("Gurbir") == 4);

assert(\*tester.search("Gurbir Randhawa") == 5);

assert(\*tester.search("Gurbalicious") == 6);

// Makes sure things are properly overwritten

tester.insert("Gurbir", 7);

assert(\*tester.search("Gurbir") == 7);

**PersonProfile**

To test Person Profile I created a test person and added in 5 attribute value pairs (simply (“1”, “2”), (“3”, “4”), etc.) and then used the GetAttVal function to make sure all the attribute value pairs were added by iterating through a loop determined by the GetNumAttValPairs function. I felt that this was sufficient testing as the code for personProfile was relatively simple, and as long as the functionality of iterating through all the attributes worked things should be fine.

**AttributeTranslator**

To test AttributeTranslator I added a line of ‘hobby, test1, hobby, test2’ to the provided translator file to make my own pair. I then loaded in the file to make sure it handled a somewhat large text file and tested a variety of random attributes as well as hobby, test1, and hobby, test2 by printing out each attribute and value in the vector of all the compatible attributes and values and making sure they matched what I manually determined by searching through the text file

**MemberDatabase**

To test MemberDatabase, I first did a test in which I loaded in a custom text file that only had a few people so I could manually check that FindMatchingMembers worked, as the provided text file had too many people to manually check. I did that by iterating through the returned vector and printing out the emails. I then loaded the provided text file to make sure that MemberDatabase would work with a very large text file, and then tested the GetMemberByEmail function by randomly selecting a few user emails and ensuring that the proper PersonProfile was returned.

**MatchMaker**

To test MatchMaker, I first did a test in which I loaded the custom text file I used to test Member Database instead of the provided text file, and manually checked that it output the correct matches. I then reverted it to loading the provided text file and then ensured that the program functioned correctly and selected a few random matches to make sure they actually matched with the passed email.