I believe I was able to implement every requirement in the spec without using any banned STL components and without any bugs.

For the RadixTree class, I tested the different cases by having the radixtree ouput every input and every new node that gets put into it. This way, I was able to check when new nodes were being created for new words and when the same path was being used for already existing nodes.

For the PersonProfile class, I created different person profiles in my main.cpp and then added different AttValPairs to different people. For each addition, I would output different things like the person’s email, or the person’s name just to make sure PersonProfile was creating the correct person. I also tested the GetAttVal function by having the function output the AttVal pair that was at the indicated index.

For the MemberDatbase class, I loaded a separate txt file that was much smaller than the original members.txt file. In doing so, I was able to efficiently debug and figure out where my errors were when using the memberDatabase class. After loading in a much smaller text file with around 3 different person profiles, I tested the FindMatchingMembers function by outputting the vector of email addresses that contained that specific AttValPair. I tested the GetMemberByEmail function by outputting the number of attvalpairs for the member with the designated email. In doing so, I was able to check whether my functions were storing/returning the correct objects.

For the AttributeTranslator class, I also loaded a separate txt file that was much smaller than the original translator.txt file. Within the load function, I outputted each pair, compatirble attribute, and compatible value to make sure I was loading in the correct information. In terms of the FindCompatibleAttValPairs function, I outputted all the compatible attribute-value pairs for the specified source attribute-value pair, which was easy to manage due to the reduced size of my new txt file.

For the MatchMaker class, I split the IdentifyRankedMatches function into three different parts and I tested each part as I progressed through the function. The first part was tested by outputting the AttValpairs for the personProfile that was newly created. I tested the second part by iterating through compatible attributes, matching attributes with collection of members that share comparible attribute-value pairs, and outputting these pairs. The third part was tested after the vector of EmailCounts was sorted and then pushed back into a separate vector based on the threshold, outputting each item within this new vector.