1. **Brief description of notable obstacles you overcame**

Overall, I felt that optimizing the program was the hardest part. At first, I didn’t pass in anything by reference or pointers. Therefore, I would obviously get stack overflows. However, by recognizing that some variables can be passed in by reference/pointers and some functions could be combined, I was able to prevent my program from stack overflowing. My final code eventually was able to handle a dictionary size of 70k in g32.

1. **Test Cases**

**Test cases to see if words with multiple anagrams are outputted properly:**

eric -> rice //Applies for when rice is inputted

art -> art, rat, tar //Applies for when rat or tar is inputted

silent -> listen, silent, tinsel //Applies for when listen or tinsel is inputted

angel -> angel, angle, glean //Applies for when angle or glean is inputted

brag -> brag, garb, grab //Applies for when garb or grab is inputted

cider -> cider, cried //Applies for when cried is inputted

elbow -> below, bowel, elbow //Applies for when below or bowel is inputted

inch -> chin, inch //Applies for when inch is inputted

save -> save, vase //Applies for when vase is inputted

cheap -> cheap, peach //Applies for when peach is inputted

point -> pinto, point //Applies for when pinto is inputted

seal -> lase, sale, seal //Applies for when lase and sale is inputted

**Test cases to see if words with no anagrams are outputted properly:**

regardless -> No matches found

blah -> No matches found

ambrosio -> No matches found

ucla -> No matches found

facebook -> No matches found

vodka -> No matches found

soju -> No matches found

**Test cases to see if words with single anagrams are outputted correctly**

10th -> 10th

nd2 -> 2nd

condanaa -> anaconda

riAmcae -> America

eric -> rice

scneeci -> science

**Test cases to make sure duplicates don’t show up**

school -> school

kool -> look

nebe -> been

add -> add, dad

gge -> egg

ozo -> zoo

**Test Cases in a small dictionary**

string results[MAXRESULTS];

string testDict[] = {"ambrosio", "ucla", "cs32", "Professor"};

int numMatches = theJumbler("broamsio", testDict, 4, results);

assert(numMatches == 1 && results[0] == "ambrosio");

string results[MAXRESULTS];

string testDict[] = {"ambrosio", "ucla", "cs32", "Professor"};

int numMatches = theJumbler("clau", testDict, 4, results);

assert(numMatches == 1 && results[0] == "ucla");

\*word with numbers\*

string results[MAXRESULTS];

string testDict[] = {"ambrosio", "ucla", "cs32", "Professor"};

int numMatches = theJumbler("32sc", testDict, 4, results);

assert(numMatches == 1 && results[0] == "cs32");

\*word with capitalization\*

string results[MAXRESULTS];

string testDict[] = {"ambrosio", "ucla", "cs32", "Professor"};

int numMatches = theJumbler("ssorefroP", testDict, 4, results);

assert(numMatches == 1 && results[0] == "Professor");

\*multiple results\*

string results[MAXRESULTS];

string testDict[] = {"ambrosio", "ucla", "clau", "cs32", "Professor"};

int numMatches = theJumbler("lauc", testDict, 4, results);

assert(numMatches == 2 && (results[0] == "ucla" || results[0] == "clau") && (results[1] == "ucla" || results[1] == "clau"));

**SAME TESTS WERE ALSO PERFORMED WITH A 100K DICTIONARY WITH UNIQUE WORDS**