Programming Assignment #3
Nicolas Benavides Levin
UIN: 127008425

Question 1:

a) q1.cpp

i) I created a for loop that iterates through the line of the sample.cpp file. First it checks for quotations, and if found, skips the characters written within quotations until it finds the second pair, and sets i to 1 past that index. I then check for //, indicating comments, and skip that line if found. I then check for special characters, and if found a potential word(identifier) is identified and inserted into the set idents. If none of the other if statements are signaled, then the string word gets appended each index of line. This string word is what is inserted into set idents.

Question 2:

a) q2.cpp

i) While the program is inside the file, it generates a new string country and an integer cases. It then finds country and cases, and sets the map coronamap's index as country and value as the number of cases for each country and case of the file. I then created an iterator that iterates through the map coronamap, and searches for the maximum amount of cases. It then sets string partOf as the country with the highest amount of cases, and couts the country with the most confirmed cases.

Question 3:

a) q3.cpp

i) insertLinear

1) I created an integer that counts the number of collisions and an integer that stores the value of x. I then created a while loop that loops until the vector "linear" at index "count" reaches -1, signaling an empty space. It increases collisions and count, and if the count becomes greater than the size of the vector "linear" it resets count to 0, essentially reiterating through the loop.

Once an empty space is found, it exits the while loops and sets linear[count] = x. I then return the number of collisions.

ii) insertQuad

1) This function works similarly to insertLinear, however I increase count using the formula (x + i + (i*i)), and increasing i through each iteration of the while loop. If count is ever greater than or equal to the size of vector quad, count gets set to (count +(i*i) -quad.size() in order to return it to a valid index within the vector. Once the while loop completes, a valid empty space is found and quad[count] = x. I then return the number of collisions.

iii) insertDuble

1) I put a return 0 at the top of my insertDuble function in order to allow my code to compile. Although my i value iterates correctly, there is an issue with iterating count when I mod the equation (x+i*dubHash) by duble.size(). The correct amount of collisions should be returning, but I continue to receive a seg fault. This may be due to my compiler, however I could not find a working solution to this error.

Question 4:

a) OrderedMap.h

i) hashFunction

1) I created a stringstream called s. I then set s' value to key. I then created a new string called strung and set its value to s.str() in order to append the values of key to string. I then created an int value called answer. I created a for loop that iterates through the string strung and then converts strung[i] to an int and adds that to int answer. Once the for loop is complete, I return answer and the Key is converted into an integer.

ii) insert

1) I created a KeyNode currKN and set it equal to KeyNode, a KeyNode prevKN and set that equal to currKN, and a new

Value Node vn that will be used later for insertion. I then created a while loop that starts at the root of currKN. First, it checks to see if the hash_key associated with currKN is equivalent to the hash_key value you are looking for. If so, the boolean isF(isFound) is set to true and you break the loop. Otherwise, you check if the current hash_key is less than the hash_key you are searching for. If so, you set prevKN = currKN and iterate currKN = currKN->down. Once this while loop breaks, your isF will either be true or false. If true, I created a ValueNode currVN and set it equal to currKN->right. I then created a while loop that ends when currVN->right reaches nullptr. Once it reaches nullptr, currVN->right = vn, AKA the new ValueNode vn is inserted. If isF is false, then a new KeyNode is made called nKN. I then set prevKN->down = nKN. Therefore all cases for insertion are complete.