PA3

Nicholas Soliman (UIN: 326009195)

Q1.) Question one asked to create a C++ file that would read an external file and output the variable names of variables used. The way that I implemented this was by first opening a file based off the users input then reading the file line by line. I would recreate a string for each line of the code and create a string array adding each word into it. Aside from words that came between quotation marks. Once the string array was created I would then compare it to the reserveArray[] given in the code. Which would eliminate any words that were not possible for variables names. Then I would output the words from the new set.

Q2.) Question two asked to create a C++ file that would read a list of country names and confirmed case numbers and find the maximum number of cases in a country. I first iterated through each line and created a variable for the country and a variables for the number of cases. Then it adds it to the map <string,int> where the string is the country and the int is the number of cases. Then after each line is read, it will iterate through the map and find the maximum number of cases in all the countries and output the number and country that has the most.

Q3.) Question three asked to create a C++ file that would have the functions insertLinear(), insertQuad(), and insertDuble(). For each of these functions it would go through a hash and insert a value but if there was already a value in that spot it would be considered a collision and the collision counter would increase by one. The way I implemented each of these was by first creating a while loop with the counter starting at 0 then it would continuously be adding up. The counter was used in reassigning the bucket value. For each function they used different things:

Linear: bucket = (bucket + constant*counter) modulo linear.size() → ax+b

Quadratic: bucket = (bucket + constant*counter + constant2*counter^2) modulo quad.size() → cx^2+ax+b

Double: bucket = (bucket + dubHash*counter) modulo duble.size()

Then I would return the total collisions in each.

Q4.) For part 1 of Question 4, I created a hash function that takes a key in as input then goes through each element of the input and creates a total value for the inputs. Then this total value was encrypted by basically multiplying, adding, and dividing a bunch of different values then modulo by the MAP_MAX_SIZE. For part 2 of the question, I created an ordered map by having KeyNodes going down the left side and ValueNodes branching off from the KeyNodes. It first creates a new “current” node then it creates a while loop that continues to run while the next key node is undefined. It then checks to see if the hashkey lines up with the hashkey of t->down. Then it shifts current down one node and creates a newValue node, which is assigned to current->right (The right of the current key node).

Then after it adds the nodes that need to be added it creates new value and key nodes and assigns the next newKey to be down one additional node.