CSCE 221 Cover Page
PA #4

First Name:Yash Kalyani UIN:827003754 Section: 511

E-mail address:yk7335@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People					
Web pages (provide URL)	https://www.techiedelight.com/convert-string-to-int-cpp/ http://www.cplusplus.com/reference/regex/regex_match/				
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Yash Kayani Date 4/13/2020

Description:

We were given the task to read in a CSV file and take in the roster which had the names of student, their UIN, email address. We then read the input file which contained the kids that had the same data inside the CSV but they also had a quiz grade .We then input this data in our data structure Hashtable which we created using Vectors with linked lists and each part of the list contained a pair with the key of the hashtable and the quiz value. We then wrote a CSV file with the entire roster including the kids with and without a quiz score.

Descriptions of data structures and algorithms used in your:

I created a Hashtable using vectors with lists inside of each vector and at each node of the list there is a std::pair structure containing the key value (index of vector (location)), and a quiz score. The hashtable took in the quiz value as a typename T. In our main we converted the string quiz grade into a integer before we inserted in the pair. The hashtable had constructor, destructor, insert, search, and other statistic algorithms. The insert function push backed a pair into a list in the index of the vector which came from the hash function. The search algorithm looks for a specific quiz grade and returns it if it's there if it isn't it returns a nullptr.

Description of input and output data. List all restrictions and assumptions that you have imposed on your input data and program:

There are two files which we read and one output file which we write too. The files we read in are the roster file and input file. The roster file contains all the students while the input file only contains the students who have a quiz score. The output file basically combines the two and gives all students and lists the score If they have one if not then that part of it is just blank. Assumptions we have made in our input data is that every data is separated by a comma and there are not any missing commas which if there were would make our output file incorrect.

Description of testing:

I tested my Statistical calculations (Min, Max, and Average length of lists) by computing it for the output roster that was constructed. The main CPP file created a output file and uses the functions inside CSV editor class and Hashtable class to construct an output CSV and fill it with values and get statistical measurements on the vector and the list. **To run the program type, make then ./main.**

Which C++ features or standard library classes have you used in your program:

#include <ostream>, #include <iostream>, #include <vector>, #include <regex>, #include <list>, #include <utility>, #include <string>, #include <stdio.h>, and using namespace std.

Provide the statistics about the hash table. Are the computational results about the hashing consistent with the expected running time for the hashing algorithm? Justify your answer:

No, the computational results are much faster than what hashing expected running time is. The average running time for inserting should be O(n) for hashing but we did ours in constant time O(1) because we used lists and chaining method as our hashing technique. Search function in hashing on average is O(n+m) and we used one for loop because we knew the exact index of the vector which made out runtime O(n). Hashing is best case O(1) and that is usually for smaller sets so in that sense our measurements were consistent with the running time of hashing algorithm in that it is Hashing's best case.

```
[yk7335]@compute ~/CSCE 2
:: ./main
M:100
minimum chain len = 0
maximum chain len = 1
average chain len = 0.17
```

Conclusion:

Creating a hashtable at first was a bit difficult because I was not quite sure how data was stored. The idea of hashtables is easier than implementing. I used online resources and TA help to finish my program. The regex part of assignment was the most challenging because it was my first time using that library, and wasn't sure what it did at first.