Assignment 1 – 3D5A – Hash Tables

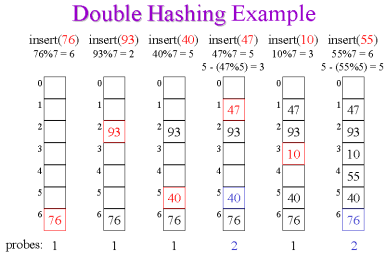
Part 1:

Originally for part 1 I had a massive hash table which had no collisions, however after the first lab I was told this was wrong and that the hash table had to be the same size as the excel file. So, I had part 1 done within a few hours originally, however with the new information it took me a lot longer to finish this section. I had many bugs which took a long time to solve, a lot of these bugs were very silly mistakes, and this meant that I couldn't spend time on section 4.
In part 1 I also was not very efficient as when I was comparing strings I did not use strcmp, I used a loop and counting which is not very accurate and takes up computing time.
In terms of collisions, I originally had 0 as I said but then when I incorporated my new algorithm I got 40 collisions.
In my storing function I had 3 different loops, which I used to tell if the space was free, if the space was not free and the word in the space was not the same and in my final loop was when the space wasn't free but the word inside the space was the same and this is when I added +1 to the counter of that name.

Part 2:

The majority of the code that I used in part 2 was the same as in part 1. First of all I had to research what it meant by a good hash function [1]. I also had to make sure that I reduced my collisions further as that Is what the assignment was about. So I thought if we have a bigger hash number and the biggest possible prime number to modulus the hash number by this would decrease the collsions. So in truth so I played around with different values and different factorials to see which method would be the best. In the end with my current method I again reduced the collisions down from 40 to 22.

Part3:
Part 3 I found hard as it took a long time to get the code to work as in so that the collisions would actually decrease instead of increasing. In the end I used the notes of blackboard and the picture from link [2] to help me with my algorithm. I choose this algorithm because it worked because I got less collisions. My collisions went from 22 down to 11.



[2]

Part4:

Part 4 I did not get very much time to spend on this due to part1 taking so long. However, I did get some to play around with the truncated file and importing this. I added more values to my struct so that it could take in all the input values. Then when I tried to import the new file I kept getting Segmentation fault and I could not get any further than this. However I will still explain my thinking behind the rest of part 4. Then when it came to allocating dynamic memory I was going to use a linked list. Where the head pointer would point to the struct array at whatever position (excuse the inaccurate code below) e.g

struct people p1[55]

*head=p1[24]

and then if another value came along and it was the same value, I would then create another node and set head->next to this new node. And continue to do this for the entire file.

Conclusions:
I got parts 1-3 working perfectly, with collisions decreasing each time however spending so much time on section 1 meant that I did not have time to complete part 4.

References

[1]https://www.sparknotes.com/cs/searching/hashtables/section2/
[2]
https://courses.cs.washington.edu/courses/cse326/00wi/handouts/lecture16/sld025.htm