Project Report

- In this project I used struct,node and thus linkedList structures.
- I defined 3 hash functions like below:

Hash1: This one does radix transformation by changing the base from 10 to 9 using the id of the student and then for it to become an index (%SIZE) is used.

Hash2: This one does shift folding by summing up the decimal values of the first 5 character of the names and then for it to become an index (%SIZE) is used.

Hash3: This one does extraction. I used 2 different variables from the student structure;id and name. Decimal value of the first 2 characters of the name and last 2 digits of the id is used. And then for it to become an index (%SIZE) is used.

- Insert function

I called one of the hash functions in here and get the index from there. I printed the index taken from the hash function in here because I couldn't do the structure properly I printed here. I checked the table I created until I find the next empty place and I added the new Node there.

- Remove function

I called one of the hash functions in here and get the index from there. With the index should be same for the same values, I simply checked the index and I used a similar method to linked list deletion with the help of 2 pointers.

- Search function

I used almost same function of the remove, I deleted the code that actually deletes the Node and instead I returned a bool variable.

- Display function

From 0 to SIZE index and all the nodes that's been inside those indexes are displayed here with nested loops.

- Utalization

Here I used the clock() function to measure the time difference before and after calling the function. I commented it out because I couldn't define the hashTemplate() function which gives me the opportunity to switch between functions, thus this one is commented it out but with one hash function only it works perfectly.

- Destroy

From index 0 to SIZE, and very similar method to linkedList deletion I destroyed whole the structure.

Time complexity of my hashing functions:

Hash1: O(logn)

Hash2:O(5) (5 iterations)

Hash3:O(1)

HashTable has O(1) for insert,remove,search functions.(average). O(n) worst case.