openhttest.cpp (Double Hashing)

- For this collision method, I made a DoubleHash.cpp file and a Person.cpp file. The DoubleHash.cpp file contains a DoubleHash class which creates a vector on instance creation. This vector stores values of data type Person. The Person object comes from class Person in Person.cpp; it contains two values: long long int number and string caller. Long long int is used because 64 bits are needed to store some phone numbers The vector is initially filled with Person objects where number = 0 and caller = "No name here.".

Destructors: No destructors are used here as almost no memory is allocated with "new".

Insert Function, O(1) Time

- When the insert(long long int k, string caller) function is called in DoubleHash, it first checks if the table is full or the key is already in there. If not, the insert() checks the value at index k%size in the vector. If the number value at this location is 0, the new number and caller are inputted. If its full, h2 is calculated based on the given formula, until a location is found where table[h1].caller is 0. The average case for this is O(1) time because when the hash table is full, "failure" is printed. This function is void.

Search Function, O(1) Time

- This function is of type string, returning the caller name upon success, and "No name associated with that number." Upon failure, which is changed to "not found" in main. When search(long long int k), is called, the function will iterate through the hash table to find the a caller name associated with that key. To allow for constant time, only h2 is calculated if the first result is not the desired number. The new vector indexes are calculated with h2, and the caller with the associated phone number is searched for. Because numbers with the same h1 value are inserted in order, as soon as "No name here." Is found, the search returns "No name associated with that number." for failure. Deleted callers are marked as "deleted", so the search knows to continue if an index is deleted and not empty.

Delete Function, O(1) Time

- This function is void. When deleteName(long long int k) is called, the function iterates through the hashtable values until the key a Table[newkey].number value. Once the value is found, the function sets the caller = "This name was deleted." and number = 0. Again, to keep this constant time, only the values in newkey = (h1+h2*i) % size are checked. Once "No name here." is found to be the caller property of Table[newkey], the function prints "failure". If a number in the table matches the key, that location is reset and "success is printed.

orderedhttest.cpp (Chaining Method)

- For this collision method, I made a Chaining.cpp, Node.cpp and LinkedList files. The Chaining.cpp file contains a Chaining class which creates a vector on instance creation. This vector stores values of data type LinkedList. The LinkedList objects are filled with an initial head of type Node in Chaining's constructor. Node has three properties: long long int number, string caller and Node *next (next is a pointer).

Destructors: A destructor is included in the LinkedList class. On the destructor call, all Node objects in the LinkedList are deallocated from memory.

Insert Function, O(1) Time

- When the insert(long long int k, string caller) function is called in Chaining, it first checks if the list at index k%size is to find duplicate keys. If the key "k" is not a duplicate, the insert() simply inserts a new Node with Node.caller=caller and Node.number = k. In order to LinkedList in table[k%size] sorted, the value of k is put into the appropriate location at the time of insertion (e.g. 2 in 1->3 is inserted in the middle of the list).

Search Function, O(1) Time

- This function is of type string, returning the caller name upon success, and "No name associated with that number." Upon failure, which is changed to "not found" in main. When search(long long int k), is called, the function will iterate through the linked list at table index k%size. If the key is not in the designated linked list, "No name associated with that number." will be returned, which signals to print "not found" in main. In the name is found, it will be returned and printed in the main method.

Delete Function, O(1) Time

- This function is void. When deleteName(long long int k) is called, the function iterates through the linked list at hash table location k%size. If the value is not found, "failure" is printed. If the value is found, that Node in the Linked List is deallocated from memory. The linked list is then reconnected with the previous Node's next value pointing to the Node after the deleted one. Upon this successful completion, "success" is printed.

Print Numbers

- This function is void and prints a list numbers. To print all the numbers in a linked list, the function simply starts from the head of the designated LinkedList at location "i" in the hash table and prints each one.

Chaining

- Search

- Delete

- Insert