P2-Design Document

1. OPEN ADDRESSING USING LINEAR PROBING:

Class Design:

I have used one class called as "openAddress". This class contains my vector hash (under private section) and all other member functions, constructors and destructors (under public section). I have declared an object h1 of this class in the main function.

Member Variabless & Member Functions:

Member variable in my class is the vector "hash" of type integer.

Member functions:

int hashIndex(int key, int m)
• Function to calculate key % size of the hash table.
• Function return type is int and the parameter type is int where the key and size of hash table are passed into the function.

Void hashSize(int m)
- Function to define the size of hash table of size m.
- Function return type is void and the parameter type is int where m is the size of hash table.

int hashInsert(int k, int n)
• Function to insert key k into the hash table.
• Function return type is int, and the parameter type is int where k is the key and n is the size of hash table.

int hashSearch(int k, int m)
• Function to search a key k in the hashtable of size m.
• Function return type is int, and the parameter type is int where k is the key and m is the size of hash table.

int hashDelete(int d, int m)
• Function to delete a key k in the hashtable of size m.
• Function return type is int and the parameter type is int where d is the key to delete and m is the size of hash table.

Constructors:

Constructor for my openAddress class is left empty as there are no parameters given. My constructor will also be initialized by default in the program.

Destructors:
Destructor in my program is used to clear the entries of the vector "hash" and clears the used memory. I used the hash. clear () function to clear the vector contents.

2. SEPARATE CHAINING:

Class Design:

I have used one class called as "hashTable" and a struct called "Node". The class contains my vector hash of type node*(under private section) and all other member functions, constructors and destructors (under public section). My struct has key and the next pointer for the linked list. I have initialized nodes to NULL that are empty or don't have a key. I have declared an object h1 of this class in the main function.

Member Variables & Member Functions:

Member variable of my class is vector "hash" of type node pointer which points to the head of the linked list. The last element in the linked list points to NULL. The member functions in my class are similar to the member functions in Open Addressing method (mentioned above).

3. CONSTANT RUNTIME IN MY PROGRAM:

Assuming uniform hashing in my design, I have achieved the average case runtime as follows:

I. Insert:
In my code, the keys are inserted into the hash table using the division method. The elements are stored in the index in such a way that the running time is constant.

II. Search:
In my code, the key that needs to be searched is not found element by element in my hash table. Instead, the division method is used again to reach the index corresponding to the key. The search continues until the index is -1(which denotes the vector index is empty). This way my search function obeys a linear running time.

III. Delete:
The search function is used to find the key that needs to be deleted from the hash table. If the key is not found in the hash table, it delivers an error condition. Else, it deletes the key in the index returned by the search function. Hence, my code doesn't search the whole hash table to find the element and delete it. Instead, the code is designed in such a way to make the running time constant.

BIBLIOGRAPHY:

https://www.sanfoundry.com/cpp-program-hash-tables-singly-linked-list/

https://www.geeksforgeeks.org/c-program-hashing-chaining/
https://www.geeksforgeeks.org/insert-a-node-at-a-specific-position-in-a-linked-list/