Date :- 01/05/2024

Name: Abinet Bushura Cholo
Net Id: ab11475

Hash function one

```cpp
unsigned long HashTable::hashCode(string word) {
    unsigned long hash = 5381; // Initial hash value
    for (size_t i = 0; i < word.length(); ++i) {
        char c = word[i];
        hash = ((hash << 5) + hash) + c; // hash * 33 + c
    }
    return hash % capacity; // Modulo to fit within the capacity of the hash table
}
```

Fig 1

Efficiency obtained by the hash function:

```
================================================
Size of HashTable               = 812745
Total Number of Collisions      = 936592
Avg. Number of Collisions/Entry = 1.2
================================================
find <word>                          : Search a word and its meanings in the dictionary.
import <path>                        : Import a dictionary file.
add <word,meaning(s),language>       : Add a word and/or its meanings (separated by ;) to the dictionary.
delTranslation <word,language>       : Delete a specific translation of a word from the dictionary.
delMeaning <word,meaning,language>   : Delete only a specific meaning of a word from the dictionary.
delWord <word>                       : Delete a word and its all translations from the dictionary.
export <language,filename>           : Export a a given language dictionary to a file.
exit                                 : Exit the program
>
```

pseudo code of the function

1. Initialize the hash value to 0.
2. Iterate over each character c in the input word.
3. For each character, update the hash value using the formula (hash * 31 + ASCII_value_of(c)) mod capacity.
   - Multiply the current hash value by 31 (a prime number commonly used in hashing).
   - Add the ASCII value of the current character c.
   - Take the modulo operation with the capacity of the hash table to ensure the hash value fits within the table's range.
4. Repeat steps 2-3 for all characters in the word.

5. Return the final hash value computed after iterating through all characters in the word.

Hash function two:

```
unsigned long HashTable::hashCode(string word) {
    unsigned long hash = 5381; // Initial hash value
    for (size_t i = 0; i < word.length(); ++i) {
        char c = word[i];
        hash = ((hash << 5) + hash) + c; // hash * 33 + c
    }
    return hash % capacity; // Modulo to fit within the capacity of the hash table
}
```

Efficiency obtained by the hash function:

```
================================================
Size of HashTable               = 812745
Total Number of Collisions      = 945111
Avg. Number of Collisions/Entry = 1.2
================================================
find <word>                          : Search a word and its meanings in the dictionary.
import <path>                        : Import a dictionary file.
add <word,meaning(s),language>       : Add a word and/or its meanings (separated by ;) to the dictionary.
delTranslation <word,language>       : Delete a specific translation of a word from the dictionary.
delMeaning <word,meaning,language>   : Delete only a specific meaning of a word from the dictionary.
delWord <word>                       : Delete a word and its all translations from the dictionary.
export <language,filename>           : Export a a given language dictionary to a file.
exit                                 : Exit the program
>
```

pseudo code of the function

1. Initialize the hash value to 5381 (initial hash value).
2. Iterate over each character c in the input word.
3. For each character, update the hash value using the formula ((hash << 5) + hash) + c, which is equivalent to hash * 33 + c.
   - Left shift the current hash value by 5 bits (equivalent to multiplying by 32) and add it to the current hash value.
   - Add the ASCII value of the current character c.
4. Repeat steps 2-3 for all characters in the word.
5. Return the final hash value computed after iterating through all characters in the word, modulo the capacity of the hash table to ensure it fits within the table's range.

Hash function three:

```cpp
unsigned long HashTable::hashCode(string str) {
    const unsigned int fnv_prime = 16777619; // FNV prime value
    const unsigned int offset_basis = 2166136261;  // FNV offset basis
    unsigned int hash = offset_basis;  // Initial hash value

    for (char c : str) {  // Iterate through the characters of the string
        hash ^= c;  // XOR the hash with the character
        hash *= fnv_prime;  // Multiply the hash by the FNV prime value
    }

    return hash%capacity;  // Modulo to fit within the capacity of the hash table
}
```

Efficiency obtained by the hash function:

```
================================================
Size of HashTable               = 812745
Total Number of Collisions      = 925586
Avg. Number of Collisions/Entry  = 1.1
================================================
find <word>                          : Search a word and its meanings in the dictionary.
import <path>                        : Import a dictionary file.
add <word,meaning(s),language>       : Add a word and/or its meanings (separated by ;) to the dictionary.
delTranslation <word,language>       : Delete a specific translation of a word from the dictionary.
delMeaning <word,meaning,language>   : Delete only a specific meaning of a word from the dictionary.
delWord <word>                       : Delete a word and its all translations from the dictionary.
export <language,filename>           : Export a a given language dictionary to a file.
exit                                 : Exit the program
>
```

pseudo code of the function

1. Initialize constants fnv_prime and offset_basis.
2. Initialize the hash value to offset_basis.
3. Iterate through each character c in the input string str.
4. For each character, XOR the current hash value with the ASCII value of the character, then multiply the hash value by the fnv_prime.
5. Return the final hash value modulo the capacity of the hash table to ensure it fits within the table's range.

Conclusion

Function two < Function one < Function Three. In terms of efficiency