Task 5

I have used linear probing to implement cache as with it, it is easier to find the next free space in hash table unlike with double hashing where finding a NULL block is less likely as compared to finding one in linear probing.

To read secrets from file and finding the corresponding message, directly from the dictionary, I first read a secret from the secretx.txt file and then looked for that secret (key) in the dictionary.txt file and if found, I returned the corresponding value (message). To make calculation slightly faster, I introduced a counter with initial value 0. When reading from dictionary file, since I knew the key to find, which was an unsigned long, I incremented the counter until its value was equal to the secret from secretx.txt and once they are equal I simply print the message.

To read secrets from file and finding the corresponding message, using cache used the similar approach. This time we make a new HashL object named cache and store the (key, value) pair in it using the FLU approach where the least frequent element is removed if the cache is full while inserting a new word. It again used the same counter approach to speed up the process.

Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Time (sec)** | *Secret 1* | *Secret 2* | *Secret 3* |
| *Without cache* |  |  |  |
| *With cache* |  |  |  |