**Task 5**

Implementation Report

The hash table for the cache chosen was the linear probing hash table, with the hash function used being polyHash() with madCompression(). The default fixed size is 1000 for the hashTable (same as the cache size variable in CacheManager class), so there is no resize function now. The block class was modified to contain a string ‘code’ (to store the secret code for every word), string ‘word’ (for the word itself), unsigned long ‘used’ - to track the number of times a word is used this is incremented every time. One member function inside the CacheManager class **decodeWithCacheMode(bool withCache, string codesFile, string decodeFile, int maxWords)** that uses one of the two decoding functions - **decodeWord\_Cache()** or **decodeWord\_Direct()** depending on which mode is selected (by the value of withCache – true/false) given for every code in the codesFile (e.g. “secret1.txt”) upto a given maximum number of words to decode (given by maxWords) and writes the word returned to the output file decodeFile (e.g. “direct-s1-decoded.txt”). Each decoding function takes a code as input and returns its corresponding code. The direct decoding function skips lines equal to the code to be found in the dictionary file to get to the same code entry in the “dictionary.txt” file. The word is read from the dictionary and returned. The cache decoding function has the first priority to lookup the code given and return the pointer to the block in memory that stores it. Every time this is done, the ‘used’ value is incremented, as the word was pulled was cache and is in more demand, then the word is returned. If the word did not already exist in the cache (NULL), then search the dictionary the same way as before but this time insert the word into the cache along with its secret code, and a default ‘used’ value of 0 initially, where the hash is computed from the secret code. If the words in cache becomes equal to the cache size (table is full), then the least frequently used block (has the code/word with least usage value) is deleted to make space for the new word, optimising the cache using LFU as instructed. There is a significant time difference (30% less approximately with cache) as seen by the screenshot between decoding the files directly and with the cache for 10,000 max words.

