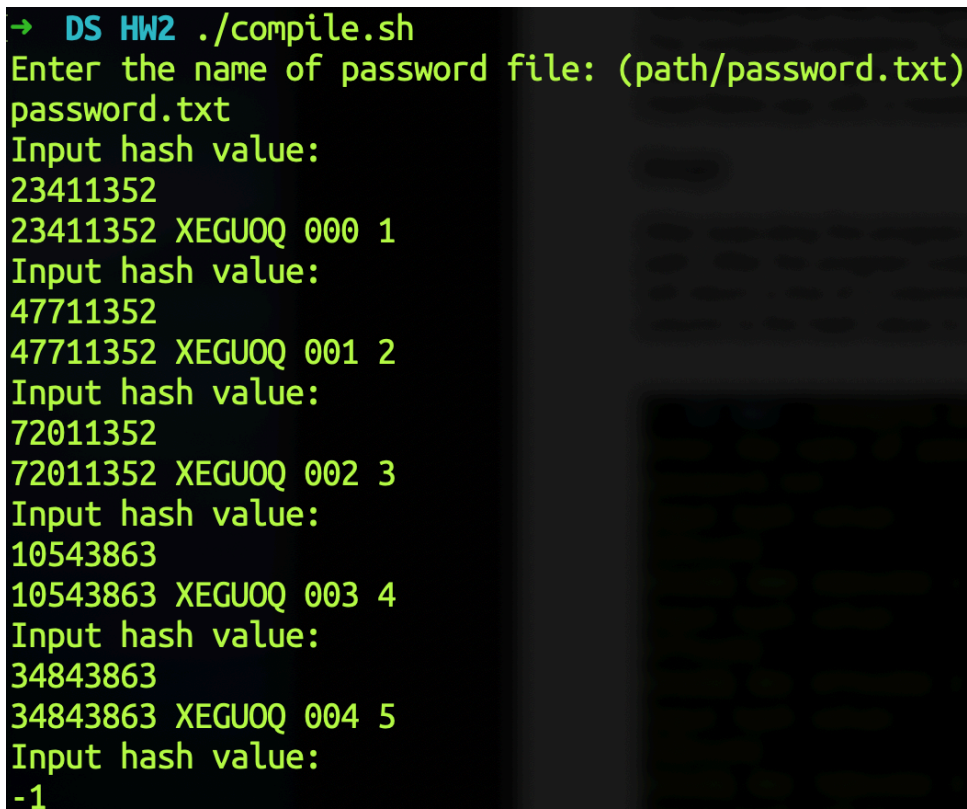


The program for the assignment is implemented by C++.

The compiled programs, FindPasswords\_User and FindPasswords, was compiled and executed by compile.sh. The C++ source code files are main.cpp, functions.cpp and HashTable.cpp with a header file, main.h.

## Usage

After executing the program, FindPasswords\_User, the user can input the name of the password.txt file with its path. After the program output, "Input hash value:", the user can input a hash value and it will return a line of 4 columns separated by a single white space character. The 1st column is the hash value to be searched. The 2nd column is the password recovered. The 3rd column is the salt corresponding to the hash value and should always contain 3 characters. The last column is the number of entries in the dictionary file that have been searched for this hash value. If the user input -1, then the program terminates.



```
→ DS HW2 ./compile.sh
Enter the name of password file: (path/password.txt)
password.txt
Input hash value:
23411352
23411352 XEGUOQ 000 1
Input hash value:
47711352
47711352 XEGUOQ 001 2
Input hash value:
72011352
72011352 XEGUOQ 002 3
Input hash value:
10543863
10543863 XEGUOQ 003 4
Input hash value:
34843863
34843863 XEGUOQ 004 5
Input hash value:
-1
```

Figure 1. The execution interface of the program, FindPasswords\_User.



```
# For batch inputs
g++ main.cpp functions.cpp HashTable.cpp -o FindPasswords;
./FindPasswords < "list_pa2.txt" > "result_pa2.txt";
```

Figure 2. The execution of the program, FindPasswords, used to generate the result\_pa2.txt automatically.

**Brief explanation of implementation****a. functions.cpp**

Most of the functions are in this file. The function, `process_file_in(string& path, Hash& hashtable)`, is the backbone of the program. It reads the passwords from the input file line by line and uses the function of Hash class to generate hash values from password strings with different salts. To make users find the corresponding passwords of hash values faster, the generated hash value, password string, the salt, and the corresponding number of entry are inserted into a hash table (Hash class) for each line.

The function, `search_pw(Hash& hashtable)`, is used to recover the password from input hash value. It is executed repeatedly, until the input is -1.

The function, `output_pwdir_file(ofstream& outfile, string& password, int& final_hash, int& salt)`, is executed during `process_file_in()` function. It outputs "Dictionary.txt" file with a password string, a salt value for each hash value.

**b. HashTable.cpp**

The class Hash declared in the header file is the data structure of hash table and its codes are in the HashTable.cpp.

The hash table is an array storing the address of items (self-defined structs). An item contains a password string, a hash value, a salt value, and a corresponding entry number. The main hash function (`hash_fun`) provided by the requirement is shown below.

$$\text{hash\_fun}(\text{string\_to\_int}) = ((243 \times \text{left\_num}) + \text{right\_num}) \bmod 85767489$$

The hash table stores items is implemented by double hashing. The hash functions used for searching are shown below.

$$\text{hash}(\text{key}) = (\text{hash}_1(\text{key}) + i \times \text{hash}_2(\text{key})) \bmod \text{size}$$

$$\text{hash}_1(\text{key}) = (1301 \times \text{key}) \bmod \text{size}$$

$$\text{hash}_2(\text{key}) = 7919 - (\text{key} \bmod 7919)$$

, where the *key* is the hash value of an input password string and *i* is the frequency of collisions.