**// Code dùng tham khảo (bài làm của sinh viên)**

**Cách 1:**

#include <iostream>

#include <string>

#include <list>

using namespace std;

#define EMPTY 0

#define DELETED -1

#define OCCUPIED 1

struct USER {

string USERNAME;

list<string> PASS;

};

struct NODE {

int flag;

USER key;

};

struct HASHTABLE {

int M;

int n;

NODE \*table;

};

int HF(HASHTABLE ht, string username) {

int s = 0;

for (int i = 0; i < username.length(); i++)

s += 37 \* username[i] + i;

return s % ht.M;

}

int HF\_LinearProbing(HASHTABLE ht, string username, int i) {

int h = HF(ht, username);

return (h + i) % ht.M;

}

void CreateEmptyHashtable(HASHTABLE &ht, int m) {

ht.M = m;

ht.table = new NODE[ht.M];

for (int i = 0; i < ht.M; i++) {

ht.table[i].flag = EMPTY;

}

ht.n = 0;

}

void Insert(HASHTABLE &ht, string username, string password) {

int i = 0;

while (i < ht.M) {

int index = HF\_LinearProbing(ht, username, i++);

if (ht.table[index].flag == EMPTY || ht.table[index].flag == DELETED) {

ht.table[index].flag = OCCUPIED;

ht.table[index].key.USERNAME = username;

ht.table[index].key.PASS.push\_back(password);

ht.n++;

return;

} else if (ht.table[index].flag == OCCUPIED && ht.table[index].key.USERNAME == username) {

ht.table[index].key.PASS.push\_back(password);

return;

}

}

}

void Traverse(HASHTABLE &ht, string username) {

int i = 0;

while (i < ht.M) {

int index = HF\_LinearProbing(ht, username, i);

if (ht.table[index].flag == EMPTY) {

cout << "Unregistered User." << endl;

return;

} else if (ht.table[index].flag == OCCUPIED && ht.table[index].key.USERNAME == username) {

for (const string &password : ht.table[index].key.PASS) {

cout << password << " ";

}

cout << endl;

return;

}

i++;

}

cout << "Unregistered User." << endl;

}

void DeleteHashtable(HASHTABLE &ht) {

delete[] ht.table;

ht.table = NULL;

ht.M = 0;

}

int main() {

int N, M;

cin >> N >> M;

HASHTABLE H;

CreateEmptyHashtable(H, 2 \* N);

string username, password;

for (int i = 0; i < N; i++) {

cin >> username >> password;

Insert(H, username, password);

}

for (int i = 0; i < M; i++) {

cin >> username;

Traverse(H, username);

}

DeleteHashtable(H);

return 0;

}