Experiment topic: **Design and Realization of Address Book Query System**

1. Experimental content

1. Description of the problem: Establish an employee address book management system for a certain unit, which can easily query the phone number and address of each employee. Design hash table storage, design and implement address book lookup system.
2. basic requirements
(1) Each record has the following data items: phone number, user name, address;
(2) Enter each record from the keyboard, and build a hash table with the phone number as the key;
(3) Use the secondary detection and re-hashing method to resolve conflicts;
(4) Find and display records for a given phone number;
(5) Save the address book information file.
(6) Design independently according to the requirements of the topic, and write a design report as required after the design is completed .

2.Code:

```
#include "stdio.h"
#include "assert.h"
#include "math.h"
#include "stdlib.h"
#include <iostream>
#include <string.h>
using namespace std;

#define SUCCESS 1
#define UNSUCCESS 0
#define SIZE 20 //Parameters that determine the size of the hash table

#define NULLKEY -65535
//---------------------------------------------- ----------------------------------//
struct data
{
	char tel[20];
	char name[20];
	char add[60];
};
```

```
typedef struct        HashTable
{
      struct data *element;//
      //int count;//
}HashTable;

int m=0;
void InitHashTable(HashTable &H);
int InsertHashKey(HashTable *H, struct data key);
int Hash_f(int key);
data SearchHashKey(HashTable* H,char num[20]);
void save(HashTable &H);

//---------------------------------------------- ----------------------------------//

int main()
{
      printf("---------------Address book system ----------------\n");
      printf("Function options:\n1. Create a new user 2. Find a user\n3. Save the address book 4.
End\nPlease choose a function:");
      int f, i; //f record input function, i is used for
      char a,number[20];
      scanf("%d",&f);
      HashTable H;
      InitHashTable(H);
      struct data elem,e;
      while(f!=4)
      {
            if(f==1)
            {

                  printf("Please enter the user name:");
                  scanf("%s",elem.name);

                  printf("Please enter the user number:");
                  scanf("%s",&elem.tel);

                  printf("Please enter user address:");
                  scanf("%s",elem. add);

                  InsertHashKey(&H,elem);
            }
            else if(f==2)
            {
```

```
            printf("Please enter the phone number of the user you want to find:");
                scanf("%s",&number);
                e=SearchHashKey(&H,number);
                if(strcmp(e.tel,"-1")==0)
                    printf("There is no such person\n");
                else
                {
                    printf("The user number: %s\n", e.tel);
                    printf("The user's name: %s\n", e.name);
                    printf("The user's address: %s\n", e.add);
                }
            }
            else if(f==3)
            {
                save(H);
            }
            printf("\n----------Contact system ----------\n");
            printf("Function options:\n1. Create a new user 2. Find a user\n3. Save the address book 4. End\nPlease choose a function:");
            scanf("%d",&f);
        }
        printf("----------Program end----------\n");
        return 0;
}

//---------------------------------------------- ----------------------------------//
void InitHashTable(HashTable &H)
{
            H.element=(data *)malloc(SIZE*(sizeof(char[20])+sizeof(char[20])+sizeof(char[60])));
        //H->count=SIZE;
            for(int i=0;i<SIZE;i++)
            strcpy(H.element[i].tel,"NULLKEY");

}
//---------------------------------------------- ----------------------------------//
//storage element
//remainder method
int InsertHashKey(HashTable *H, data key)
{
        int key2=0,i;
        for(i=0;key.tel[i]!='\0';i++)
            key2=key2+key.tel[i]-48;
        int addr=Hash_f(key2);
        while(strcmp(H->element[addr].tel,"NULLKEY")) //Conflict
```

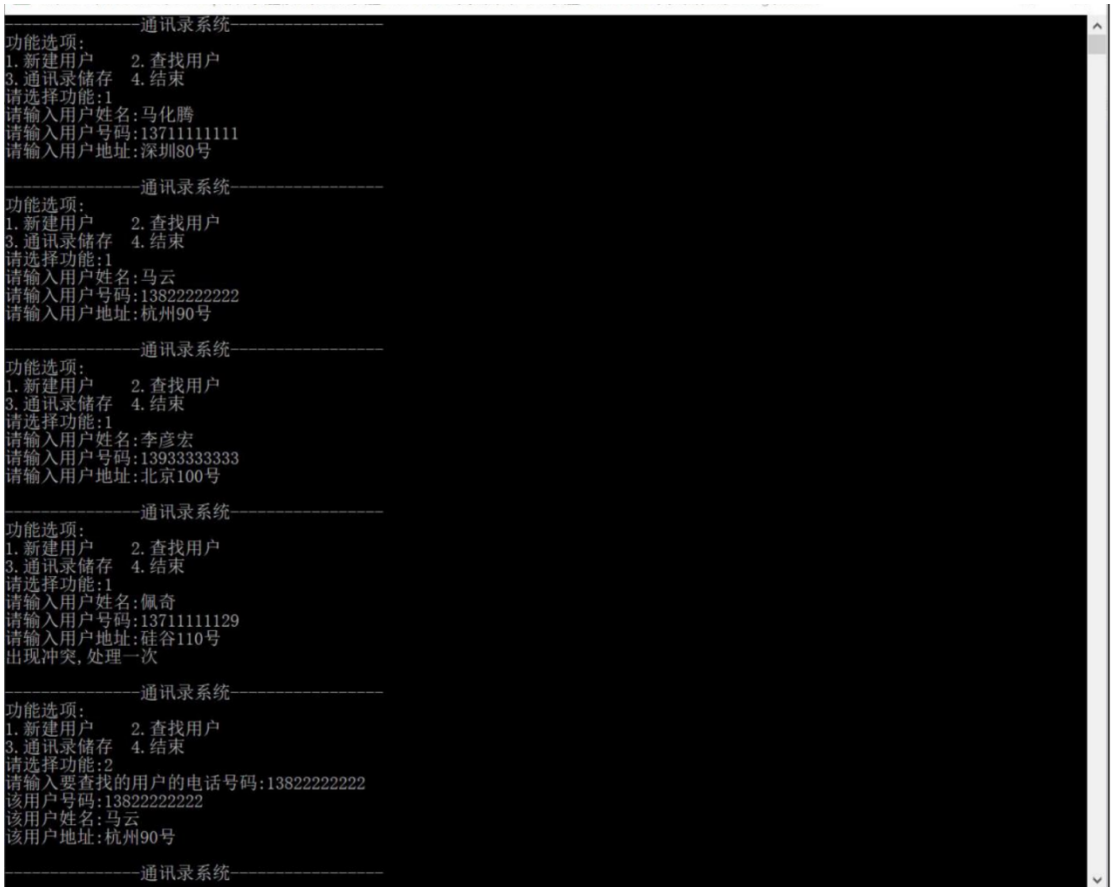
```
	{
		printf("There is a conflict, deal with it once\n");
		addr=(addr+1)%SIZE;//Resolve conflicts
	}
	strcpy(H->element[addr].add,key.add);
	strcpy(H->element[addr].name,key.name);
	strcpy(H->element[addr].tel,key.tel);
	return 0;
}
//---------------------------------------------- ----------------------------------//
//find element
data SearchHashKey(HashTable* H,char num[20])
{
	data k;
	int key2=0,i;
	for(i=0;num[i]!='\0';i++)
		key2=key2+num[i]-48;
	int addr=Hash_f(key2);
	while( strcmp(H->element[addr].tel,num) )
	{
		addr=(addr+1)%SIZE;
		if(strcmp(H->element[addr].tel,"NULLKEY")==0 || addr==Hash_f(key2))
		{strcpy(k.tel,"-1"); return k;}//UNSUCCESS
	}
	strcpy(k.tel,H->element[addr].tel);
	strcpy(k.name,H->element[addr].name);
	strcpy(k.add,H->element[addr].add);
	return k;//The search is successful, and the storage location is returned
}
//---------------------------------------------- ----------------------------------//
int Hash_f(int key)
{
	return key%SIZE;
}

void save(HashTable &H)
{
	int k=1; //record user serial number
	FILE *fp;
	//Open the file Textfile.txt
	fp=fopen("List.txt","w");
	if(fp==NULL)
	{
		printf("create file failed\n");
```

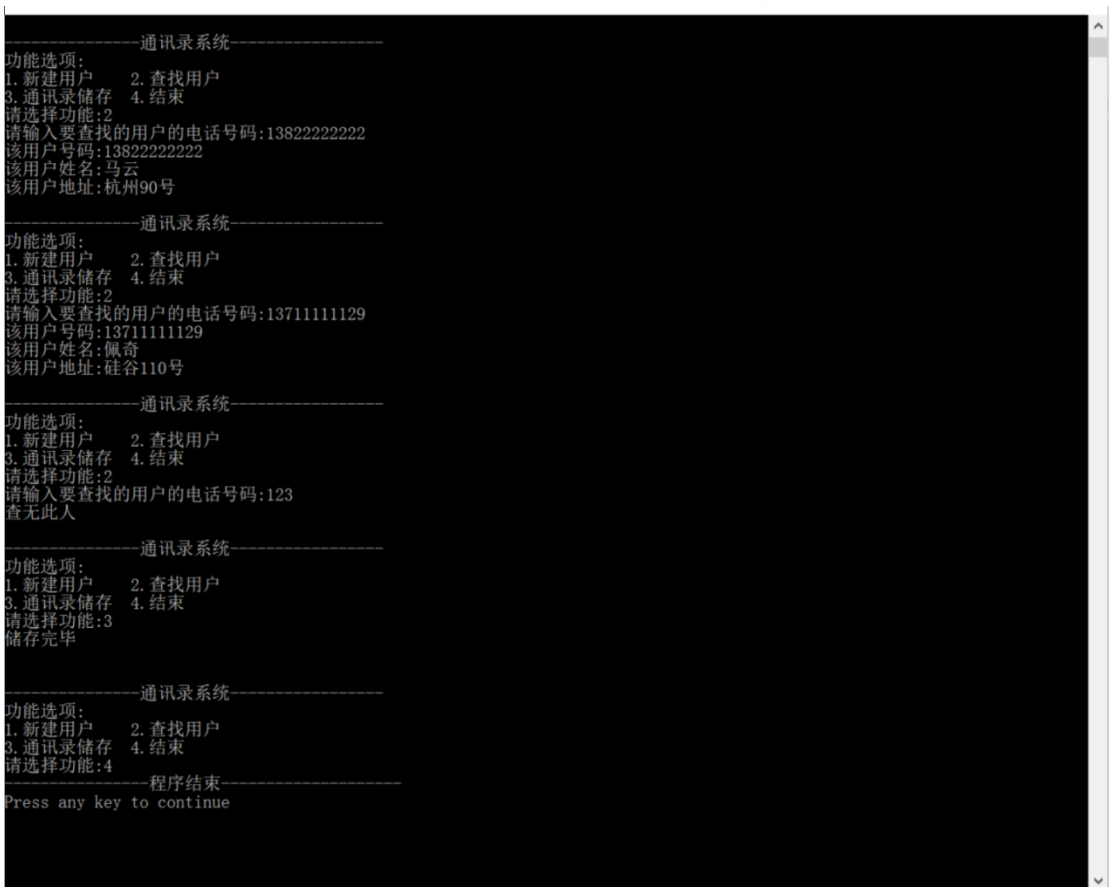
```
    }
    //Store the decoding result
    for(int i=0;i<SIZE;i++)
    {
        if(strcmp(H.element[i].tel,"NULLKEY")!=0)
        {
        fprintf(fp,"User%d:\n",k);k++;
        fprintf(fp,"User hash address location:%d\n",i);
        fprintf(fp,"User Name:%s\n",H.element[i].name);
        fprintf(fp,"user number:%s\n",H.element[i].tel);
        fprintf(fp,"User Address: %s\n\n",H.element[i].add);
        }
    }
    fclose(fp);
    printf("Save complete\n");
    printf("\n");
}
```

Four: Experimental test

The screenshot of the experiment is as follows:

As shown in the figure below, 4 functions are designed in this experiment. Function 1 Create a new user, automatically create and process the hash table, and the Page item entered in the figure below simulates the conflict and displays it. Function 2 can accurately output conflicting and non-conflicting information. Function 3 automatically saves the address book into txt text, and the text storage result is the same as that shown in the attachment, and the address location of each hash user is given in the attachment for experimental reference.


----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:1
请输入用户姓名:马化腾
请输入用户号码:13711111111
请输入用户地址:深圳80号
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:1
请输入用户姓名:马云
请输入用户号码:13822222222
请输入用户地址:杭州90号
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:1
请输入用户姓名:李彦宏
请输入用户号码:13933333333
请输入用户地址:北京100号
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:1
请输入用户姓名:佩奇
请输入用户号码:13711111129
请输入用户地址:硅谷110号
出现冲突,处理一次
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:2
请输入要查找的用户的电话号码:13822222222
该用户号码:13822222222
该用户姓名:马云
该用户地址:杭州90号
----------通讯录系统----------

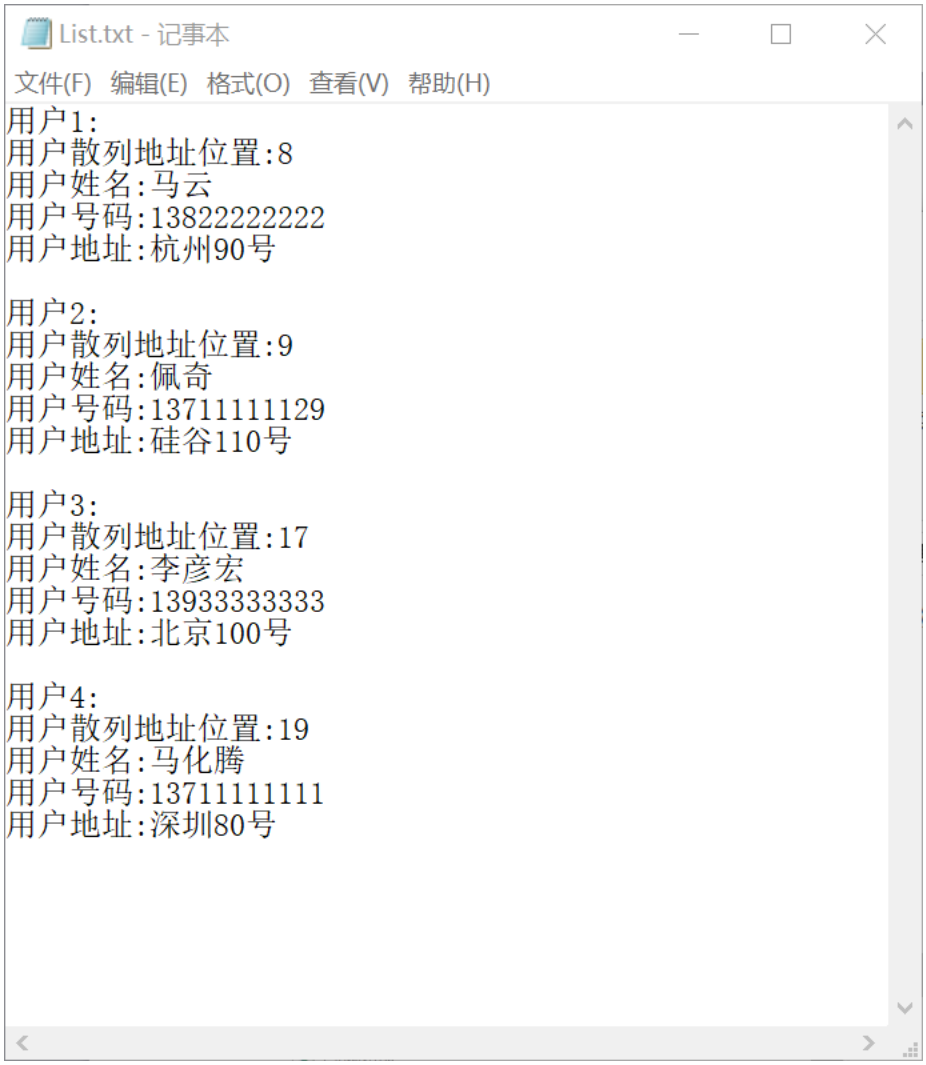

----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:2
请输入要查找的用户的电话号码:13822222222
该用户号码:13822222222
该用户姓名:马云
该用户地址:杭州90号
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:2
请输入要查找的用户的电话号码:13711111129
该用户号码:13711111129
该用户姓名:佩奇
该用户地址:硅谷110号
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:2
请输入要查找的用户的电话号码:123
查无此人
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:3
储存完毕
----------通讯录系统----------
功能选项:
1.新建用户 2.查找用户
3.通讯录储存 4.结束
请选择功能:4
----------程序结束----------
Press any key to continue

List.txt:
User 1:
User hash address location: 8
Username: Jack Ma
User number: 13822222222
User address: No. 90, Hangzhou

User 2:
User hash address location: 9
Username: Paige
User number: 13711111129
User Address: Silicon Valley 110

User 3:
User hash address location: 17
User Name: Robin Li
User number: 13933333333
User address: Beijing 100

User 4:
User hash address location: 19
Username: Ma Huateng
User number: 13711111111
User address: No. 80, Shenzhen


List.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
用户1:
用户散列地址位置:8
用户姓名:马云
用户号码:13822222222
用户地址:杭州90号
用户2:
用户散列地址位置:9
用户姓名:佩奇
用户号码:13711111129
用户地址:硅谷110号
用户3:
用户散列地址位置:17
用户姓名:李彦宏
用户号码:13933333333
用户地址:北京100号
用户4:
用户散列地址位置:19
用户姓名:马化腾
用户号码:13711111111
用户地址:深圳80号