CS 146 Programing Assignment 4

Micro-Version of Facebook Using Hashing with Chaining

Yu Xiu

Advisor: Dr. Mike Wu

Dec. 16, 2018

Table of Contents

ILLUSTRATION OF THIS MICRO-VERSION OF FACEBOOK

In this Micro-Version of Facebook project, the purpose is to simulate the functions of Facebook including add a friend, search a friend, list a person's friend and delete a friend of a person. In this project, the idea that I applied is using linked list data structure and hash table (hash map) data structure. Hash table is efficient and functional when delete an element or insert an element, since each element associated with an index. By finding the index, which created from a hash function, the element would be quickly located.

DESIGN:

This project provides command line interface. Initially, the person's name and his or her friends were packed in a **Person** class, where the friends are stored in a linked list data structure. Then create a generic class name **myLinkedList** which contains linked list data structure's functions. At last, creating a **myHashTable** class which contains the hash table methods. In this class, making the hash table as an ArrayList which stores a Person object as a linked list (Linked list stores Person object). To convert each String name of a person, using the following formula:

[cn*128^(n-1) + cn-1* 128^(n-2) + … +c1* 128^0] % m

where cn…c1 are ASCII code for each corresponding character, and m as the next prime number of the number of key entries * 1.3, and it is not too close to 2^p. It means, if I have 30 key entries, m = 41, which is the next prime number of 39 since 30*1.3 = 39.

For example, "Kate"-> [75 * 128^3 + 97 * 128^2 + 116 * 128 + 101] % 41
= [[(75 * 128 + 97) * 128 + 116] * 128 + 101] % 41
= [[[((75 * 128 + 97) % 41) * 128 + 116] % 41] * 128 + 101] % 41
= [[(21*128+116) % 41] * 128 + 101] % 41
= [16 * 128 + 101] % 41
=17

A LIST OF CLASSES AND FUNCTION CALLS AND EXPLANATIONS OF EACH PURPOSE

CLASS 1: TESTHASHING

TestHashing.java: This class contains main function and the list of the persons' names and their friends'. In this class the testing results of creating a table with all the persons and their friends, deleting a person from friend list, adding a friend, and checking whether two persons are friends. And also, it provides options for users to choose which function of this Micro-Version Facebook the users would like to test.

FUNCTIONS AND EXPLANATION:

----

1. **main(String[] args)**: This function is the main function. It contains a while loop that allow users to choose an option and test specific function of Micro-Version Facebook.

CLASS 2: PERSON

Person.java: This class contains two methods Person(String name), which is a constructor and toString(), which represents the object as a string.

FUNCTIONS AND EXPLANATION:

----

1. Person(String name): Constructor; initialize the object's state.
2. toString(): It returns the string representation of the object.

CLASS 3: MYLINKEDLIST

myLinkedList.java: This generic linked list class contains the methods that would be used to implement functions of delete a node of a linked list. And also, it provides the function to add a node in the list. List of functions: addLast(E n), addFirst(E n), contains(E x); remove(E x); toString();

FUNCTIONS AND EXPLANATION:

----

1. **class Node<e>:** A Node class which store the node value and next pointer. It contains a constructor Node(E n) to initialize value;
2. **addLast(E n):** Add a node at the end of the list.
3. **addFirst(E n):** Add a node at the head of the list.
4. **Contains(E x):** Check whether a string contained in the list.
5. **remove(E x):** Remove a node out of the list.
6. **toString():** Represent the object as a string.

CLASS 4: MYHASHTABLE

myHashTable.java: This class provides the structure of hash table. It defines the size of table, which decided by using the next prime of key entry numbers * 1.3, and it is not too close to 2^p. It also provides hash function to convert string to number. And also, it provides searching a key, delete a friend, add a friend and create a person functions.

FUNCTIONS AND EXPLANATION:

----

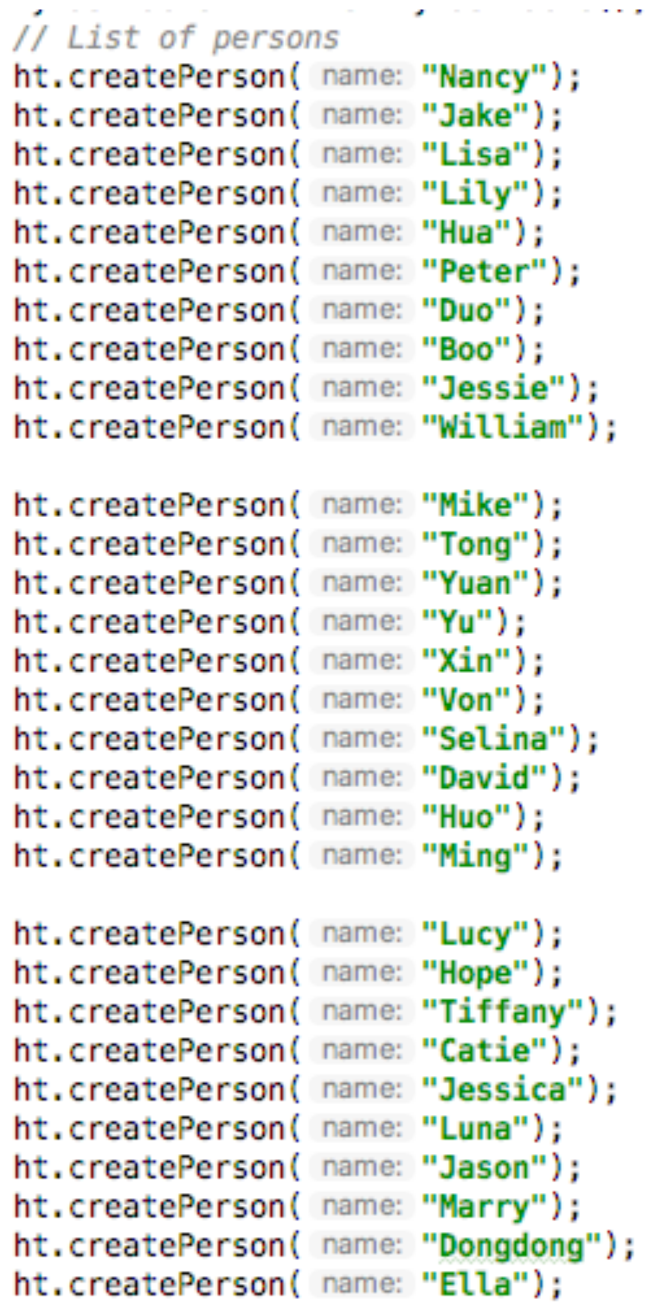
1. myHashTable(): Constructor.
2. hashCode(String key): It converts name string to an integer hash code to help to map to the slot of table.
3. isEmpty(): Check empty.
4. createPerson(String name): Create a person record of the specific name. Using the hasdCode method to convert a name to a hash code, and check whether the mapped slot is empty. If yes, directly map key to the slot, otherwise, insert the key as the head of the linked list of that slot.
5. makeFriends(String name1, String name2): Check whether two persons are friends, if not, make the two persons friends by adding each of them into their friends lists mutually.
6. chainedHashSearch(String name): Given a name, search a Person object, which contains the person's friends as well.

7. removeFriend(String name): Given a name, remove the person in the friend list. If a person has that friend, remove him mutually to the person.
8. listFriends(String name): List a person's friends.
9. checkFriends(String name1, String name2): Check whether two persons are friends.
10. toString(): String representation of object.

SCREEN SHOTS OF EACH SIMULATION PROCESS

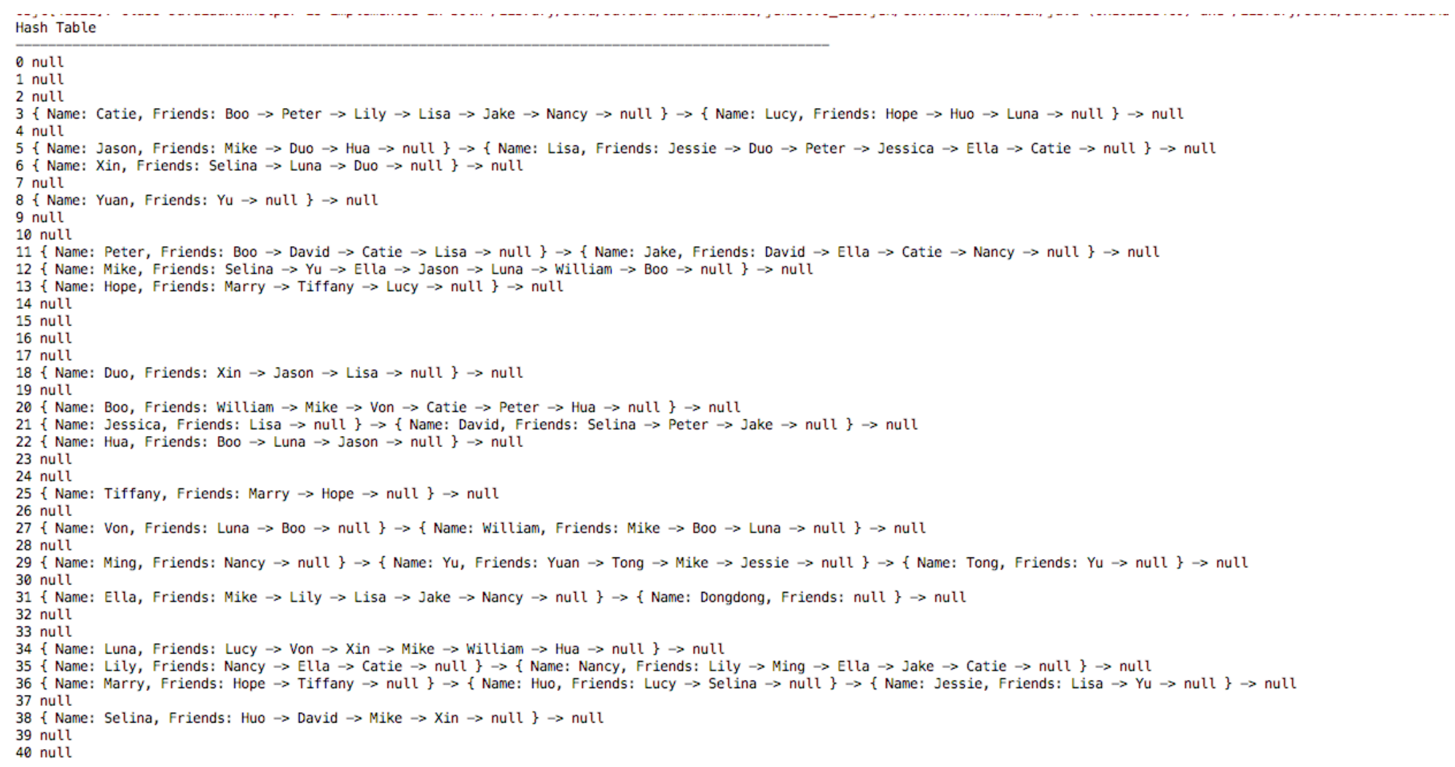
Screen Shots:

A LIST OF PERSONS:



```
// List of persons
ht.createPerson( name: "Nancy");
ht.createPerson( name: "Jake");
ht.createPerson( name: "Lisa");
ht.createPerson( name: "Lily");
ht.createPerson( name: "Hua");
ht.createPerson( name: "Peter");
ht.createPerson( name: "Duo");
ht.createPerson( name: "Boo");
ht.createPerson( name: "Jessie");
ht.createPerson( name: "William");

ht.createPerson( name: "Mike");
ht.createPerson( name: "Tong");
ht.createPerson( name: "Yuan");
ht.createPerson( name: "Yu");
ht.createPerson( name: "Xin");
ht.createPerson( name: "Von");
ht.createPerson( name: "Selina");
ht.createPerson( name: "David");
ht.createPerson( name: "Huo");
ht.createPerson( name: "Ming");

ht.createPerson( name: "Lucy");
ht.createPerson( name: "Hope");
ht.createPerson( name: "Tiffany");
ht.createPerson( name: "Catie");
ht.createPerson( name: "Jessica");
ht.createPerson( name: "Luna");
ht.createPerson( name: "Jason");
ht.createPerson( name: "Marry");
ht.createPerson( name: "Dongdong");
ht.createPerson( name: "Ella");
```

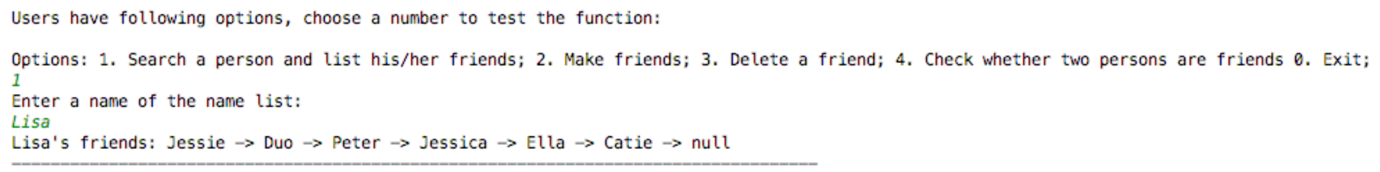
CREATE A TABLE WITH RECORD PERSON:



```
Hash Table
------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> Nancy -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> David -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Peter -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Mike -> Lily -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Nancy -> Ella -> Catie -> null } -> { Name: Nancy, Friends: Lily -> Ming -> Ella -> Jake -> Catie -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```

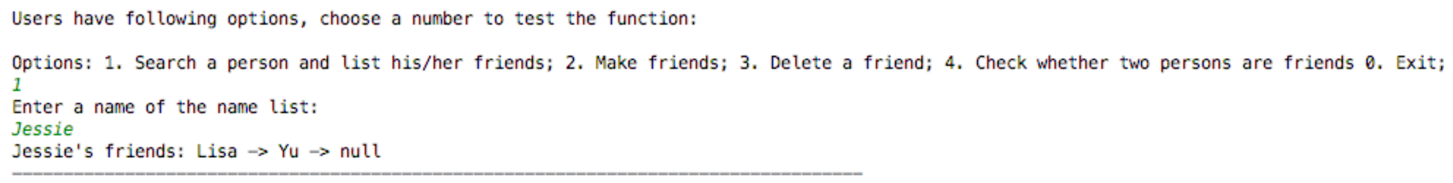
SEARCH A PERSON AND LIST HIS/HER FRIENDS:

Input: Lisa:



```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friend; 4. Check whether two persons are friends 0. Exit;
1
Enter a name of the name list:
Lisa
Lisa's friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null
------------------------------------------------------------------------------------------
```

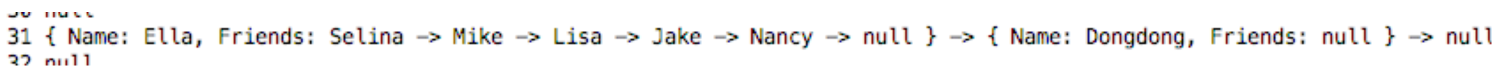
Input: Jessie:



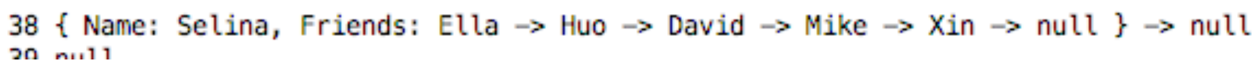
```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friend; 4. Check whether two persons are friends 0. Exit;
1
Enter a name of the name list:
Jessie
Jessie's friends: Lisa -> Yu -> null
------------------------------------------------------------------------------------------
```

MAKE FRIENDS:

Input: Ella, Selina:

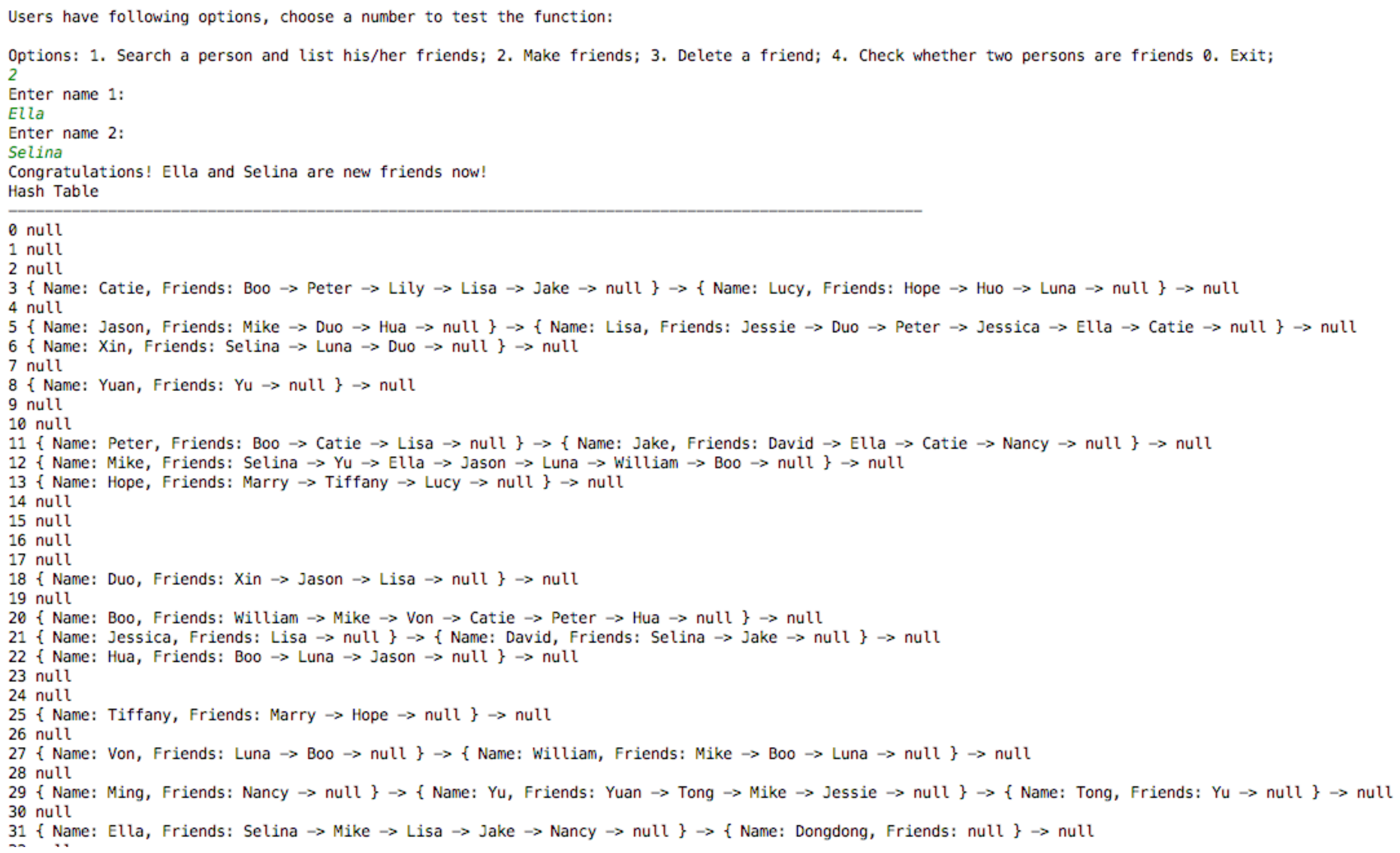


```
31 { Name: Ella, Friends: Selina -> Mike -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
```

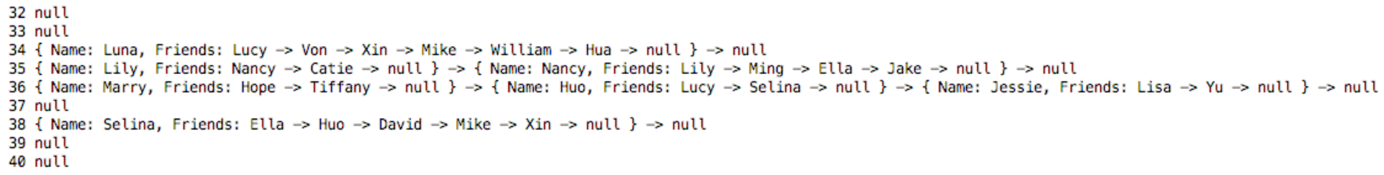


```
38 { Name: Selina, Friends: Ella -> Huo -> David -> Mike -> Xin -> null } -> null
```

The whole table after making friends of Ella and Selina:



```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friend; 4. Check whether two persons are friends 0. Exit;
2
Enter name 1:
Ella
Enter name 2:
Selina
Congratulations! Ella and Selina are new friends now!
Hash Table
----------------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Selina -> Mike -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
```

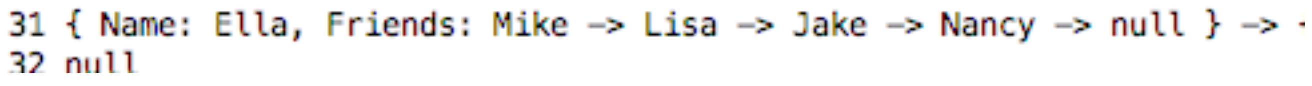


```
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Nancy -> Catie -> null } -> { Name: Nancy, Friends: Lily -> Ming -> Ella -> Jake -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Ella -> Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```

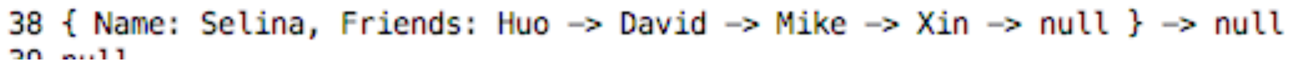
Delete/Unfriends a Person from the Friend List:

Input: Ella, Selina:

Ella does not have Selina in friends list, so as Selina:

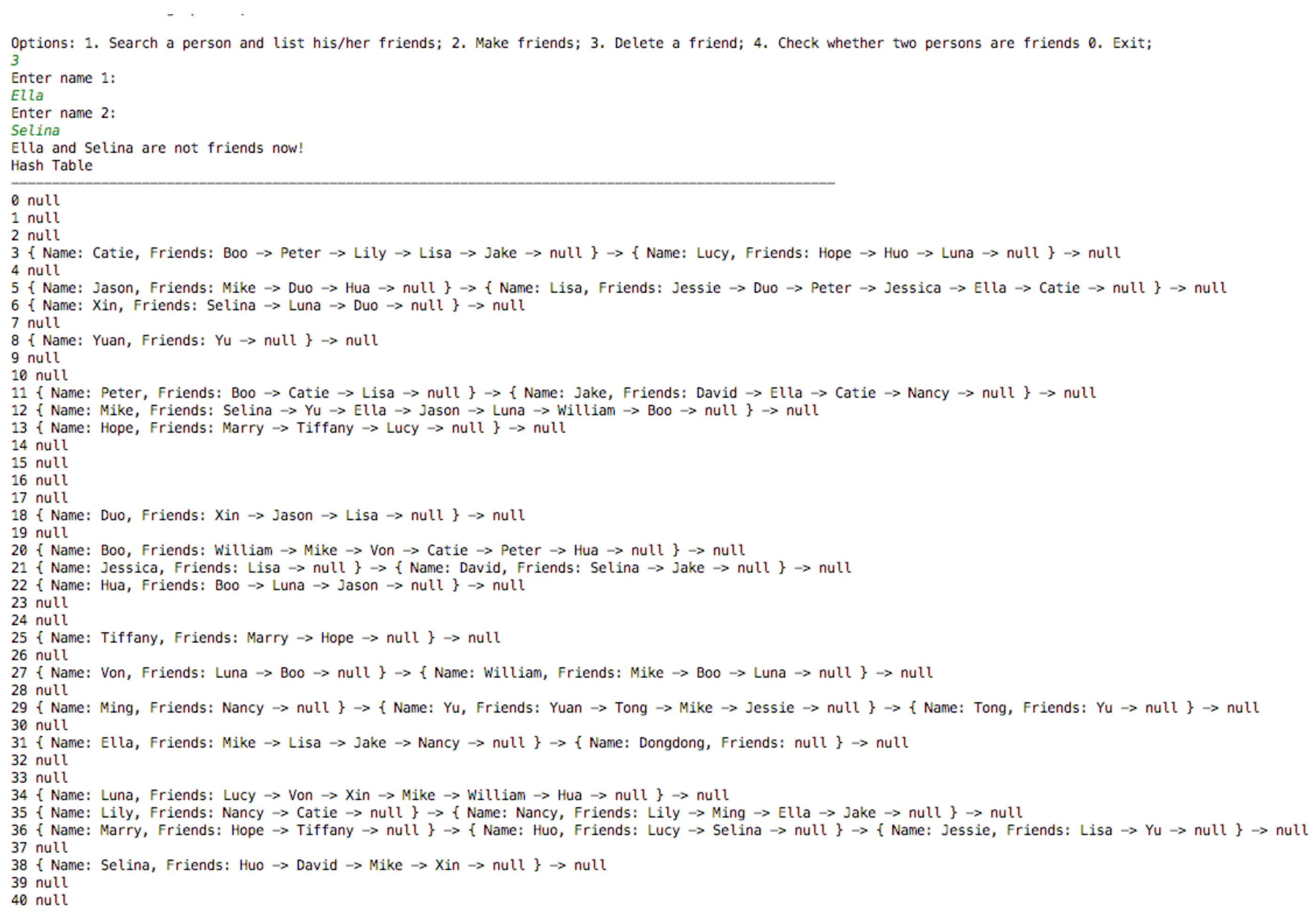


```
31 { Name: Ella, Friends: Mike -> Lisa -> Jake -> Nancy -> null } ->
32 null
```



```
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
```

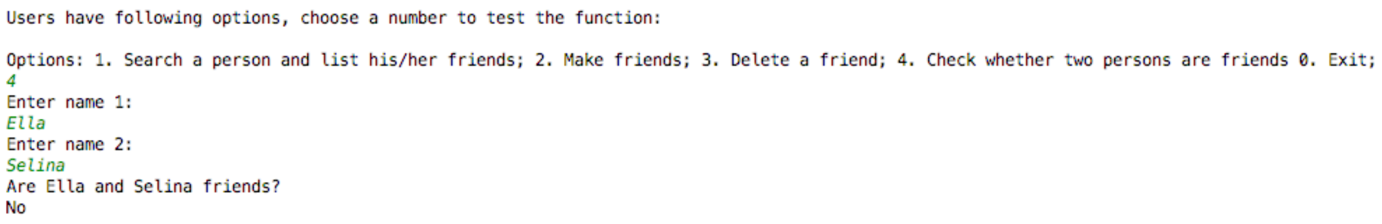
Whole table after unfriend Ella and Selina:



```
Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friend; 4. Check whether two persons are friends 0. Exit;
3
Enter name 1:
Ella
Enter name 2:
Selina
Ella and Selina are not friends now!
Hash Table
----------------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Mike -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Nancy -> Catie -> null } -> { Name: Nancy, Friends: Lily -> Ming -> Ella -> Jake -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```

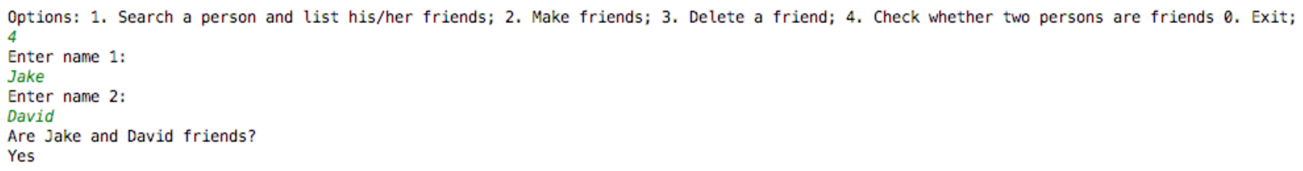
Check whether two Persons are Friends:

Input: Ella, Selina:



```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friend; 4. Check whether two persons are friends 0. Exit;
4
Enter name 1:
Ella
Enter name 2:
Selina
Are Ella and Selina friends?
No
```

Input: Jake, David:



```
Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friend; 4. Check whether two persons are friends 0. Exit;
4
Enter name 1:
Jake
Enter name 2:
David
Are Jake and David friends?
Yes
```

THE PROCEDURE OF HOW TO UNZIP FILES, INSTALL APPLICATION, AND RUN CODES

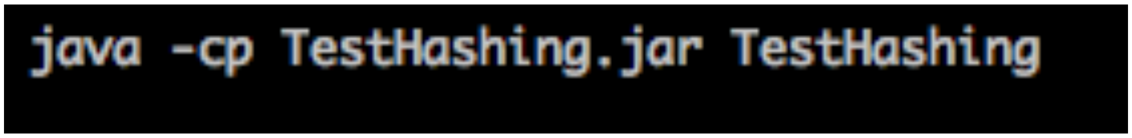
YU-XIU-PA4.ZIP

```
Yu-Xiu-PA4/
  Yu-Xiu-PA4-Report.pdf: This is the assignment report.
  TestHashing.jar: This is the jar file of my TestHashing project.
  TestHashing/
     src/
      main/
        java/
          myHashTable.java
          myLinkedList.java
          Person.java
          TestHashing.java
          META-INF/
            MANIFEST
```

RUN THE JAR FILE IN CONSOLE:

java –cp TestHashing.jar TestHashing

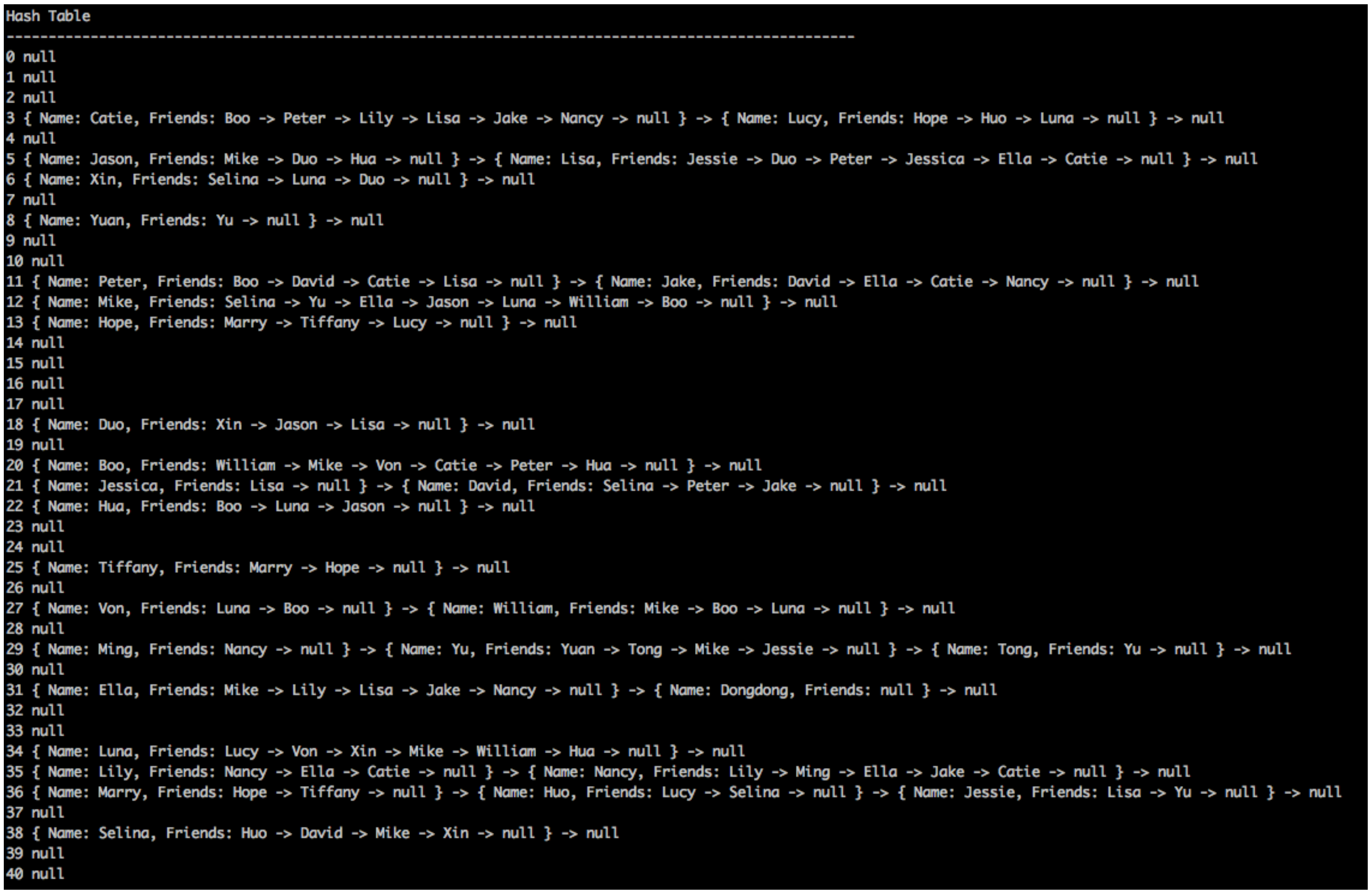
screen shot:



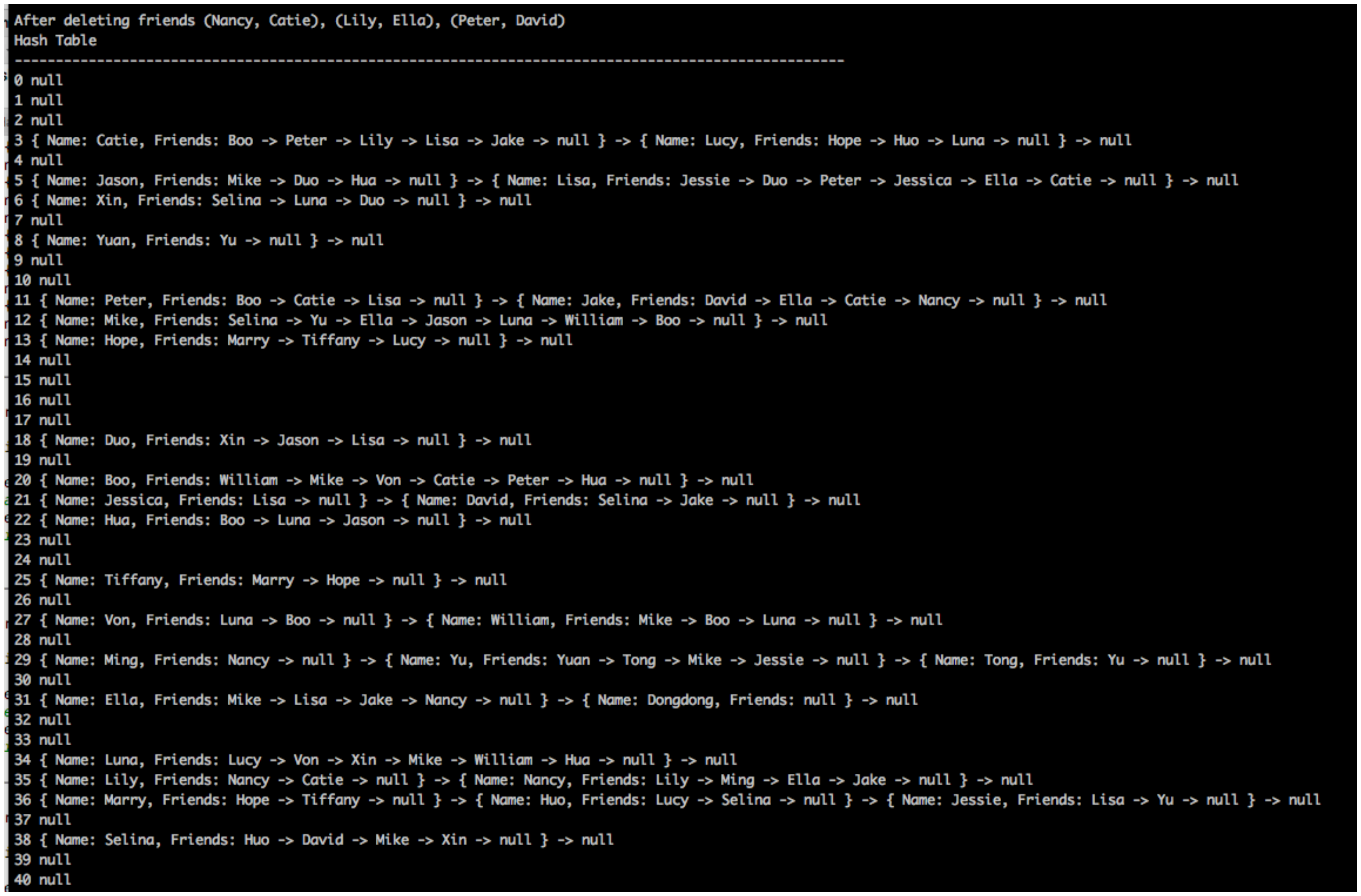
```
java -cp TestHashing.jar TestHashing
```

PROCEDURE OF HOW TO RUN MY CODES:

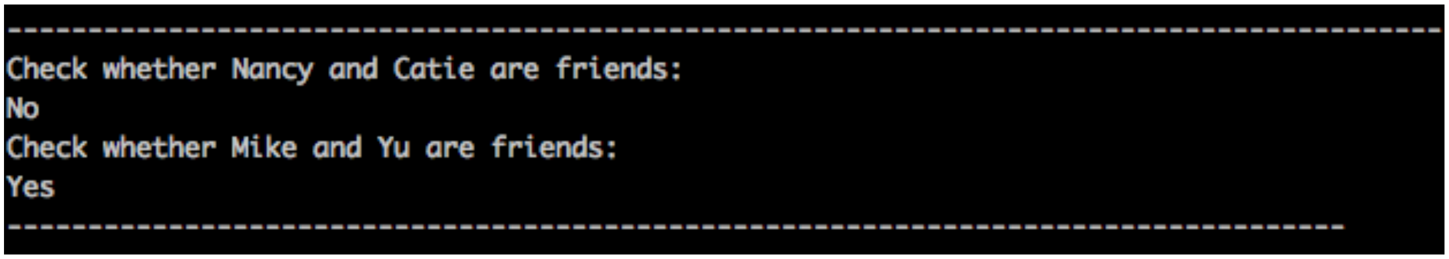
When running the file in console, user would see the Hash Table with the list of persons' names and friends, a table of deleting friends, and a check whether two persons are friends, and then the user would be provide options to test the funtions:



```
Hash Table
----------------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> Nancy -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> David -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Peter -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Mike -> Lily -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Nancy -> Ella -> Catie -> null } -> { Name: Nancy, Friends: Lily -> Ming -> Ella -> Jake -> Catie -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```

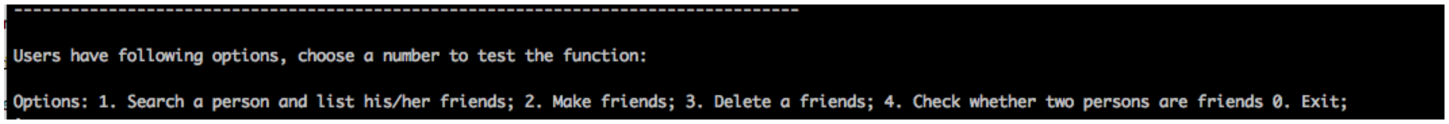


```
After deleting friends (Nancy, Catie), (Lily, Ella), (Peter, David)
Hash Table
----------------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Mike -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Nancy -> Catie -> null } -> { Name: Nancy, Friends: Lily -> Ming -> Ella -> Jake -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```



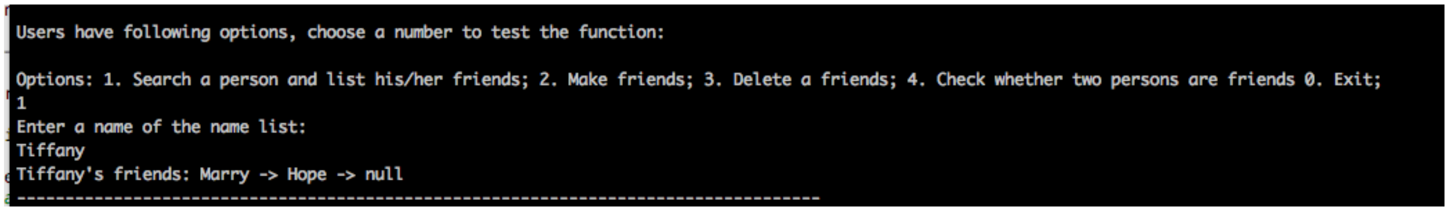
```
-----------------------------------------------------------------------------------
Check whether Nancy and Catie are friends:
No
Check whether Mike and Yu are friends:
Yes
-----------------------------------------------------------------------------
```

After that, users would be provided options to test the functions of Micro-Version Facebook



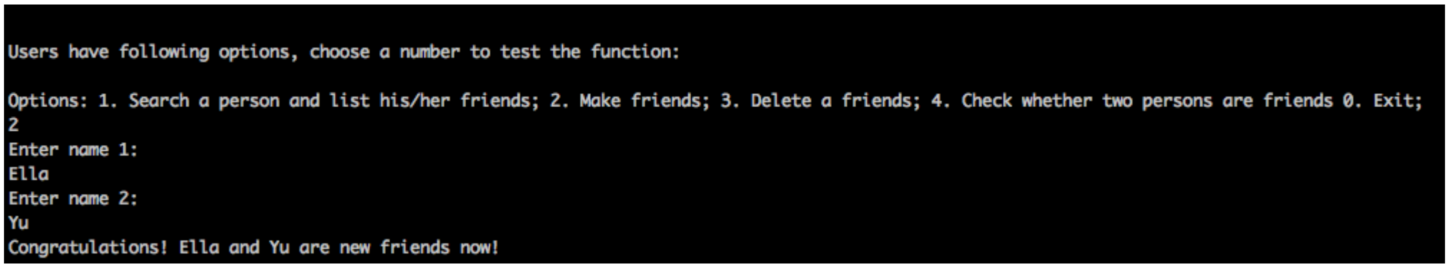
```
--------------------------------------------------------------------------------

Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friends; 4. Check whether two persons are friends 0. Exit;
```

Enter **1**: User could search a person and list the person’s friends:

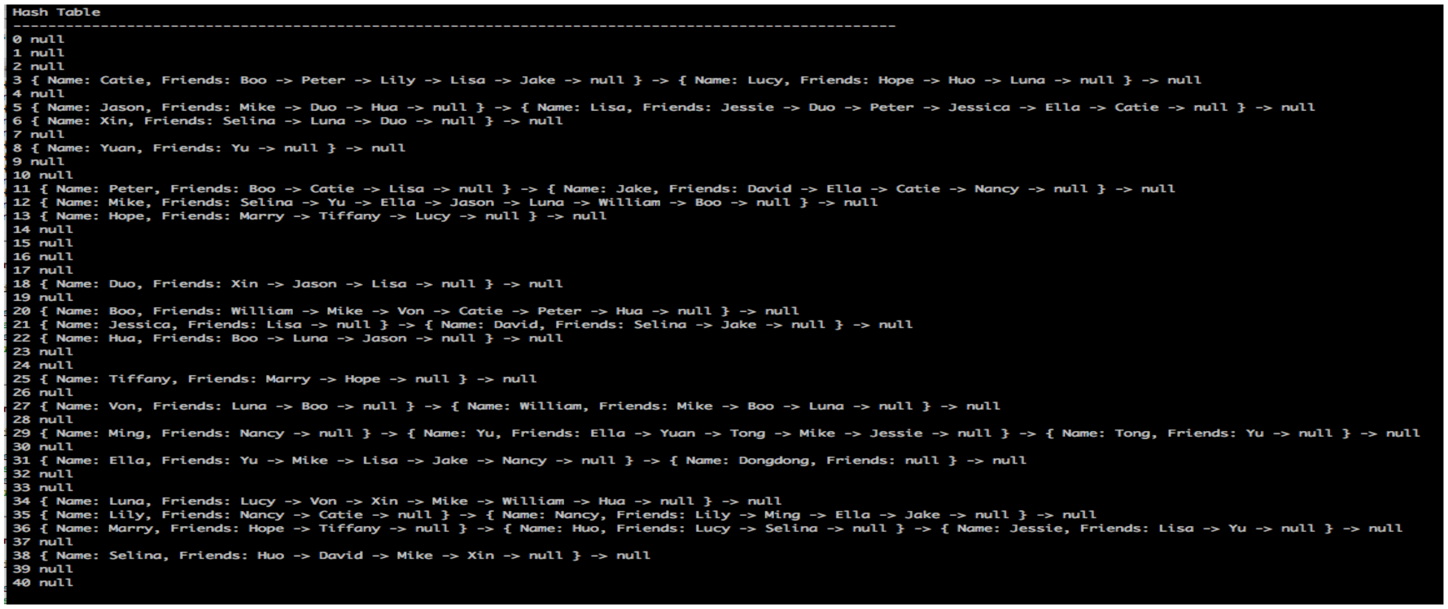


```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friends; 4. Check whether two persons are friends 0. Exit;
1
Enter a name of the name list:
Tiffany
Tiffany's friends: Marry -> Hope -> null
--------------------------------------------------------------------------------
```

Enter **2**: Test making friends of two people and display the table

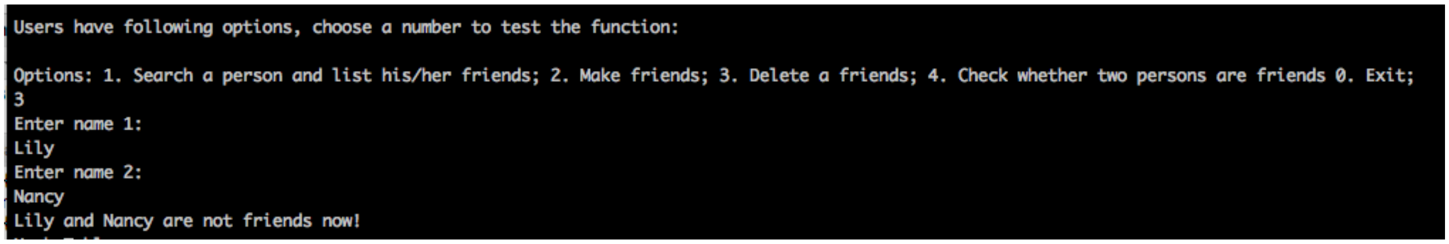


```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friends; 4. Check whether two persons are friends 0. Exit;
2
Enter name 1:
Ella
Enter name 2:
Yu
Congratulations! Ella and Yu are new friends now!
```

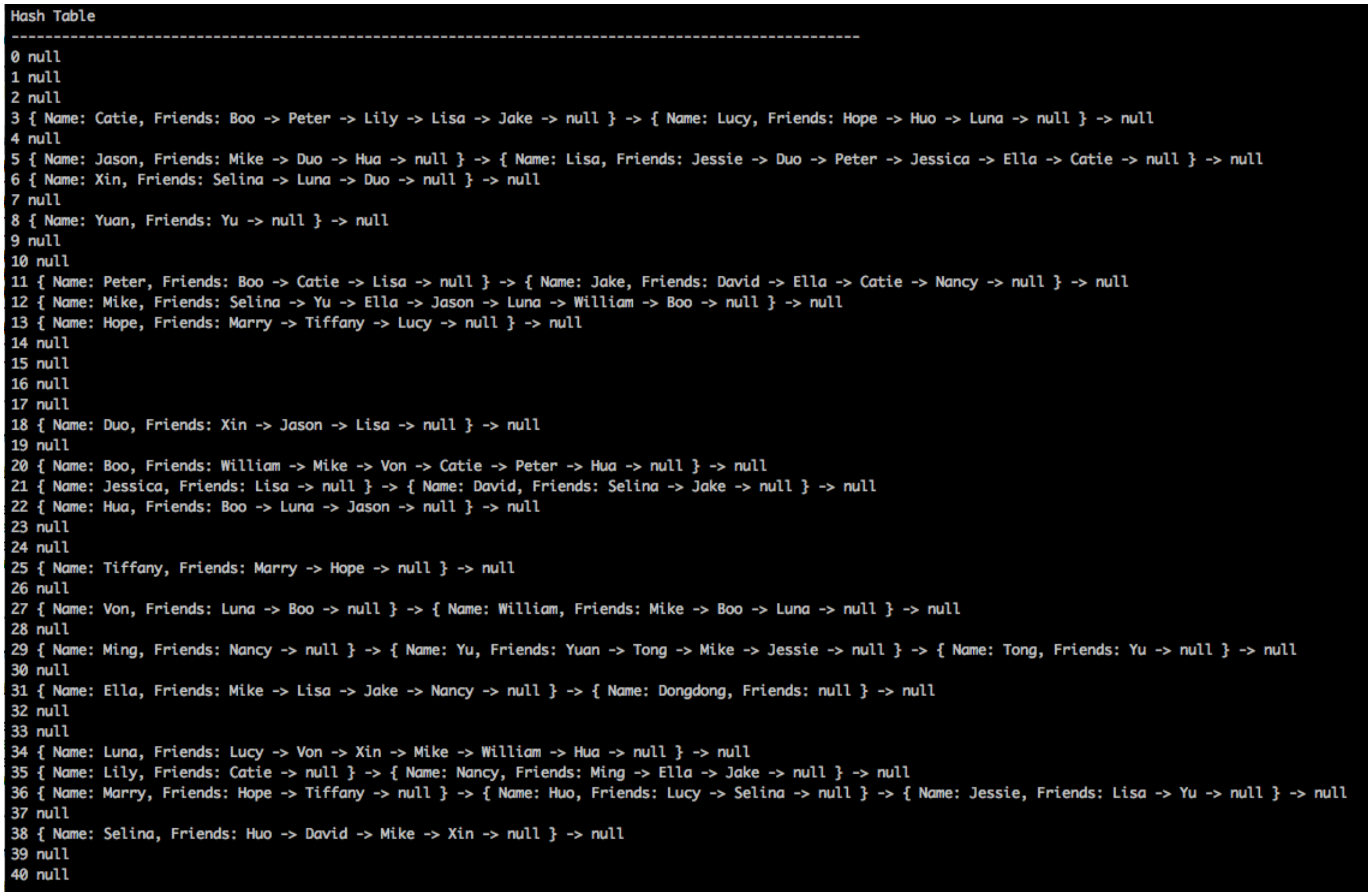


```
Hash Table
----------------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Ella -> Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Yu -> Mike -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Nancy -> Catie -> null } -> { Name: Nancy, Friends: Lily -> Ming -> Ella -> Jake -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```

Enter **3**: Test deleting friends of a person and display the table

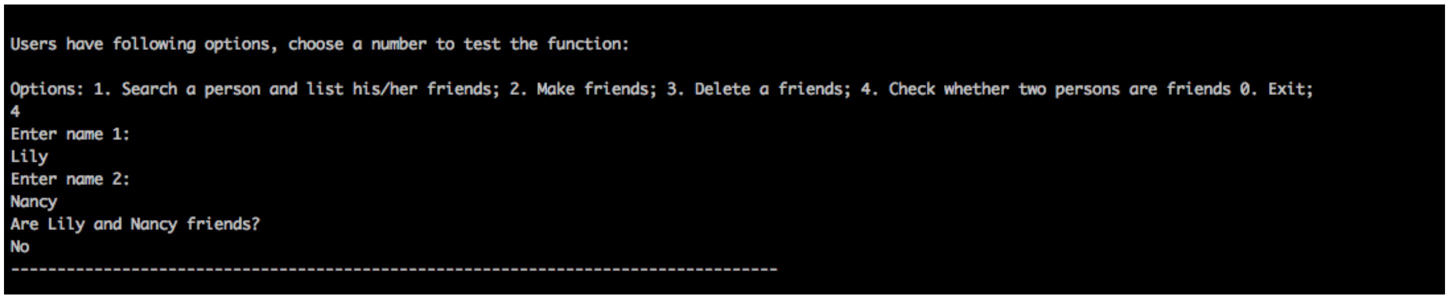


```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friends; 4. Check whether two persons are friends 0. Exit;
3
Enter name 1:
Lily
Enter name 2:
Nancy
Lily and Nancy are not friends now!
```



```
Hash Table
----------------------------------------------------------------------------------------------------
0 null
1 null
2 null
3 { Name: Catie, Friends: Boo -> Peter -> Lily -> Lisa -> Jake -> null } -> { Name: Lucy, Friends: Hope -> Huo -> Luna -> null } -> null
4 null
5 { Name: Jason, Friends: Mike -> Duo -> Hua -> null } -> { Name: Lisa, Friends: Jessie -> Duo -> Peter -> Jessica -> Ella -> Catie -> null } -> null
6 { Name: Xin, Friends: Selina -> Luna -> Duo -> null } -> null
7 null
8 { Name: Yuan, Friends: Yu -> null } -> null
9 null
10 null
11 { Name: Peter, Friends: Boo -> Catie -> Lisa -> null } -> { Name: Jake, Friends: David -> Ella -> Catie -> Nancy -> null } -> null
12 { Name: Mike, Friends: Selina -> Yu -> Ella -> Jason -> Luna -> William -> Boo -> null } -> null
13 { Name: Hope, Friends: Marry -> Tiffany -> Lucy -> null } -> null
14 null
15 null
16 null
17 null
18 { Name: Duo, Friends: Xin -> Jason -> Lisa -> null } -> null
19 null
20 { Name: Boo, Friends: William -> Mike -> Von -> Catie -> Peter -> Hua -> null } -> null
21 { Name: Jessica, Friends: Lisa -> null } -> { Name: David, Friends: Selina -> Jake -> null } -> null
22 { Name: Hua, Friends: Boo -> Luna -> Jason -> null } -> null
23 null
24 null
25 { Name: Tiffany, Friends: Marry -> Hope -> null } -> null
26 null
27 { Name: Von, Friends: Luna -> Boo -> null } -> { Name: William, Friends: Mike -> Boo -> Luna -> null } -> null
28 null
29 { Name: Ming, Friends: Nancy -> null } -> { Name: Yu, Friends: Yuan -> Tong -> Mike -> Jessie -> null } -> { Name: Tong, Friends: Yu -> null } -> null
30 null
31 { Name: Ella, Friends: Mike -> Lisa -> Jake -> Nancy -> null } -> { Name: Dongdong, Friends: null } -> null
32 null
33 null
34 { Name: Luna, Friends: Lucy -> Von -> Xin -> Mike -> William -> Hua -> null } -> null
35 { Name: Lily, Friends: Catie -> null } -> { Name: Nancy, Friends: Ming -> Ella -> Jake -> null } -> null
36 { Name: Marry, Friends: Hope -> Tiffany -> null } -> { Name: Huo, Friends: Lucy -> Selina -> null } -> { Name: Jessie, Friends: Lisa -> Yu -> null } -> null
37 null
38 { Name: Selina, Friends: Huo -> David -> Mike -> Xin -> null } -> null
39 null
40 null
```

Enter **4**: Test whether two persons are friends



```
Users have following options, choose a number to test the function:

Options: 1. Search a person and list his/her friends; 2. Make friends; 3. Delete a friends; 4. Check whether two persons are friends 0. Exit;
4
Enter name 1:
Lily
Enter name 2:
Nancy
Are Lily and Nancy friends?
No
-------------------------------------------------------------------------------------
```

Enter **0**: Exit the program

PROBLEMS ENCOUNTERED DURING THE IMPLEMENTATION.

1. To create a functional has function required a lot thinking.
2. To convert strings of names to a number and mod it by a prime number m requires is hard. If I use all the characters and their ASCII codes, when multiply each code with 128^p and add all them up, it would be quickly out of the boundary and the number might be too large, so that I use mathematics to allocate modulo operations.

   For example, "Kate"-> [75 * 128^3 + 97 * 128^2 + 116 * 128 + 101] % m
   = [[(75 * 128 + 97) * 128 + 116] * 128 + 101] % m
   = [[[((75 * 128 + 97) % m) * 128 + 116] % m] * 128 + 101] % m
3. It was confusing at the beginning that the friend list of a person and the Hash table chaining all use LinkedList.

LESSONS LEARNED

1. I learned how to implement Hashing with chaining.
2. I learned how to implement chaining with LinkedList.
3. I learned how to find a functional hash function to map keys into a slot of the table array.
4. I liked how Facebook works with hash tables to make a friend and delete a friend.