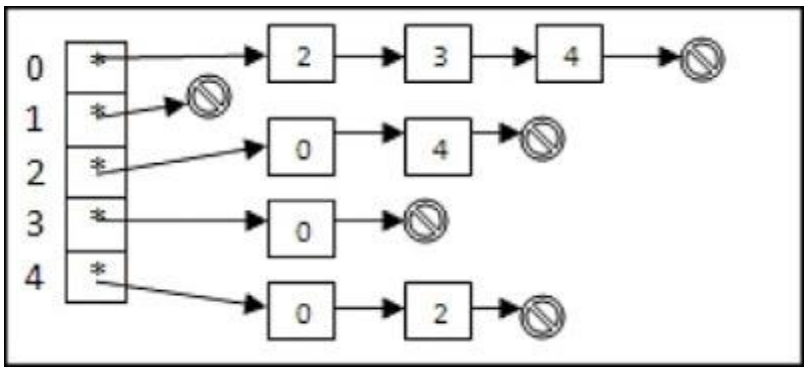
Implement a class called Fakebook, which is a fake Facebook, with the following data members (more members can be added if required):

```
class Fakebook{
private:

     List allusers;

     node** friendInfo; //an array of pointers to the heads of linked lists
     int n;             //number of users

public:

     //the methods go here
};
```

The array **`friendInfo`**, contains pointers to linked lists, each of which contain pointers to nodes of users who are friends of the user represented by that index of the array. For example, look at the following **`friendInfo`** structure:



This Fakebook contains five users, with ids: 0, 1, 2, 3 and 4, such that, 0 is the friend of 2, 3 and 4; 1 is nobody's friend; 2 is the friend of 0 and 4; 3 is the friend of 0, and 4 is the friend of zero and 2.

At any stage during the execution of the program the Fakebook must be maintained correctly. That is, the friend information must be correctly maintained.

Now, will see what the list actually looks like in this case. For this we will have to decide what the node looks like. A node is defined by the following class:

```
class node{
     User * puser;
     node* next;
     //we will need to make getters setters in public interface
};
```

And **`User`** itself is the following class:

```
class User
{
Private:
     string name;
     int id; //this is the id shown in the picture //above string
location;
     bool gender;
     int age;
};
```

So each **`node`** in the linked lists of the Fakebook contains a pointer to a dynamically allocated user.

Now, following is the set of operations we have to add to the Fakebook.

Load Users and create Fakebook

From a file called "allusers.txt" we have to load the users' information. This file contains lines like the following

Umar Ali#Lahore#M#24
John Erik#Berlin#M#22
Zainab Omer#Islamabad#F#19
…

Our program will have to read this file line by line, create a user dynamically for each line and add its pointer to the List called **`allusers`** present in the Fakebook class. This should all be done in a member function of the Fakebook called***loadUsers***. Remember that we should give id=0 to the user on first line, id=1 to the user on the second line, and so on.

Up till this point the array pointer **`node** friendInfo`** is null.

Load friend information for Fakebook

From a file called “friends.txt” we have to load the friends’ information. This file contains lines like the following:

2, 1, 9, 4, 3
3, 2, 1, 4, 5
…

This means that user with Id=2 is friends with users with ids = 1, 9, 4 and 3, and user with id=3 is friends with users with ids=2, 1, 4 and 5 etc.

Using this information our program (a member function called***populateFriendInfo***) will go through the list **`allusers`** which is already populated and use the pointers in that list to populate the **`friendInfo`** structure. REMEMBER: WE MUST NOT ALLOCATE THE USERS AGAIN. WE SHOULD USE POINTERS TO THE SAME USERS THAT HAVE ALREADY BEEN CREATED IN THE LIST **`allusers`**. SO IF THERE ARE TOTAL OF n USERS THE PROGRAM NEVER ALLOCATES MORE THAN n **`User`** OBJECTS DYNAMICALLY.

Now that the Fakebook has been populated, we have to provide the following member functions:

1) printFakebook

It will print the names of all user along with their friends count and other information about Fakebook. For example, following is a possible output of this function:

Users: 90 Males: 40 Females: 50

Johnny Davis (Chicago), Male, 22 yrs old, 4 friends

Sarah Alice (Berlin), Female, 16 yrs old, 2 friends

…

2) addUser(const User&obj)

It will add a new user to the **`allusers`** list (make sure it has a new id) and also, in the **`friendInfo`** structure, set its index pointer to null (because it has no friends to start with).

3) addFriend(int id1,int id2)

It will take ids to two users and make them friends of each other.

4) removeFriend(int id1,int id2)

It will take ids to two user and unfriend them.

5) deleteUser(int id)

It will delete the user with the given id from `allusers`, and also update the`friendInfo` structure.

6) listAllFriends(string name)

It will print the names of all the friends of the user with the given name.

7) listAllMutualFriends(string name1, string name2)

It will print all the mutual friends of the users with names name1 and name2.

8) showThreeMostLikelyPeopleIknow(string name1)

It will print the names of three people who have the greatest number of mutual friends with the user with this name1, but who are not themselves friends of name1 (this is one way in which the actual Facebook suggests you new friends).

9) printAllUsersBetweenAges(int agelow, int agehigh)

Prints the users between ages given in parameters.

10) printAllMalesBetweenAges(int agelow, int agehigh)

Prints the male users between ages given in parameters.

11) printAllFemalesBetweenAges(int agelow, int agehigh)

Prints the female users between ages given in parameters.

12) Constructors and Destructor