**Project**

After your graduation from UMT, you have been hired by Facebook as a Software engineer. On the first day of your job, you are required to write a C++ program for the following problem:

You are given an input file (**friends.txt**) containing data about friendship status of some persons. The first line in the input file contains a positive integer (you can assume that ) indicating the number of persons in a small social network. The second line of the input file contains another positive integer ***F*** (indicating the number of friendships). After that there are ***F*** lines, each line containing a pair of integers indicating that person is friends with the person **j**.

***Example input file:* friends.txt**

**6**

**7**

**0,1**

**1,2**

**3,1**

**1,5**

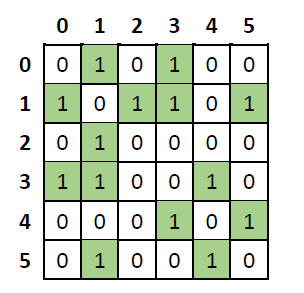
**4,5**

**3,0**

**4,3**

First of all, you are required to store the friendships in the social network using a two-dimensional array of **short**s. In the case of above sample input file, you can visualize the friendships in the form a array. Your code must be **generic** i.e. it must work for all values of **N** in the range For this, you can define a two-dimensional array **50 x 50** of **shorts**, and use only the first **N** rows and first **N** columns of this two-dimensional array depending upon the value of read from the input file.

Note that, friendship is a two-way phenomenon. Each pair means that: person **i** is friend with person **j** and also that person **j** is friend with person **i**. So, each pair will cause two entries in the two-dimensional friendship array to be set to **1**. For example, the friendships given in the above example text file will be stored like this:



Also note that all entries on the *main diagonal* (which runs from top-left to bottom-right corner) will always be ZERO i.e. a person cannot be friend with himself/herself.

**The flow of your program will be as follows:**

1. Read the data from the input file, and store it in a two-dimensional array (as described above).
2. Display the friendship table on screen, in a neat and readable way. All rows and columns should be properly labeled in the output.
3. Repeat the following logic until the user chooses to quit the program:
   1. Ask the user to enter two different integers (let’s say **A** and **B**). Perform **input validation** to make sure that Aand **B** are NOT same, and both are valid i.e. and.
   2. ***Note:*** You MUST use more meaningful variable names.
   3. Your program should determine and display the count and list of all persons which are COMMON FRIENDS of persons **A** and **B**. See the following sample run to understand the working of this step.

Sample run for **Step # 3** in the above list is shown below. Note that this sample run corresponds to the example input file shown on the previous page. Text shown in **red** in entered by the user:

**......**

**......**

**Enter two integers: 1 4**

**Person # 1 and 4 have 2 common friend(s) which is/are: 3 5**

**Continue (Y/N)? Y**

**Enter two integers: 4 0**

**Person # 4 and 0 have 1 common friend(s) which is/are: 3**

**Continue (Y/N)? Y**

**Enter two integers: 1 3**

**Person # 1 and 3 have 1 common friend(s) which is/are: 0**

**Continue (Y/N)? Y**

**Enter two integers: 4 5**

**Person # 4 and 5 have NO common friend(s)**

**Continue (Y/N)? N**

**Important Note:** You MUST implement the logic of your program using at least **4 different functions** (apart from the **main** function).