Homework #1

Overview:

In this assignment, you will prepare a C++ program that uses lists to solve a problem. In many large companies, there are socials for the new hires that involve "collecting" handshakes or other interactions from the other participants. We would like to identify the order that participants finish interacting with all other participants.

For example, consider a group of four participants: A, B, C, and D.

The interactions, in order are:

1. A B
2. A C
3. B D
4. B C
5. A D
6. C D

Given this ordered set of pairings, we conclude that participant B is the first to interact with all of the other participants. In interaction 1, A and B meet. In interaction 3, B and D meet. Finally, B and C meet in interaction 4. At this point, B has met all other participants, while A has not met D, C has not met D and D has not met A or C.

The order that B has met the other participants is: [A, D, C].

Requirements:

The purpose of the program is to identify the order that each participant finishes meeting all the other participants and report the order of interactions with the other participants. In particular, we are interested in identifying the first three people who meet all the other participants. Output the first person to meet all the other participants on the first line. Then, output the second person to meet all the other participants on the second line. Finally, output the third person to meet all the other participants on the third line. **In the case of a tie between two participants, output the first person listed in the meeting first.**

Make good use of functions and classes to implement the Lists necessary to solve this problem. You may use an array-based or link-list based implementation to keep track of the interactions. Your program should read the requested file, using the ArgumentManager, and output to the requested file.

All input files will start with a single integer on the first line that indicates the number of participants. Each subsequent line will contain a pair of values that represent the interaction between two participants. **These values can be integers or strings** and are separated by a single space. The entries are always valid, and always consist of only two valid participants per line. Participants in each file are either strings or integers, not both. **You may not use Vectors**. You should use the List implementations you have created for lab.

Input File:

5

2 3

3 4

0 3

0 1

0 4

1 4

1 2

1 3

2 4

0 2

Output File:

1: 1 -> 0, 4, 2, 3

2: 3 -> 2, 4, 0, 1

3: 4 -> 3, 0, 1, 2

In this example, there are 5 participants numbered 0 to 4. They interact in pairs as shown in the input file. The first person to meet everyone is participant 1. The order participants met is listed after the arrow. For 1, this is 0, 4, 2, 3. The next line lists the second participant to finish, person 3. For 3 the meetings in order are 2, 4, 0, and then 1. The next line shows that 4 finished third, meeting 3, 0, 1, and then 2. These interactions are enumerated and separated by a comma and space, on one line, in the output file. This means that the solution should provide a means to track the interactions and the order they occur.

Note: A pairing can occur in either order and a pairing between the same two participants can appear multiple times in a file. Only consider the first pairing of any two participants

IMPORTANT: **Homework is individual**. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. Code that is copied from another student (for instance, renaming variables, changing for and while loops, changing indentation, etc.) will be detected and result in a **0** for your homework grade. The limit is 80% similarity.

Homework 1 must be submitted to the class Linux server.

Homework #1

Make sure to create a folder under your root directory, and name it "hw1" (case sensitive), copy any .cpp and .h file to this folder, "ArgumentManager.h" needs to be included as well.

Be sure that your file names do not include spaces, or the testing script will fail.

You do not need to include any of the input, output, or answer files into the folder, but you may wish to upload these files for testing purposes.

Be sure to grant permission to the folders for grading.