CSCI-2270
Assignment 9 XC [OPTIONAL]
Instructor: Boese

Social Networks

Friends connected to friends and their hobbies

Objectives

- Add and traverse a user-built graph with vertices and edges
- Use a simple hashing function for a hashtable with pointers to vertices in a graph

In the comment block at the top of your program list your full name, date, assignment as Social Network and include whether you want this to replace your lowest assignment score or your lowest quiz score (your only 2 options).

Data Files

All the data files for a given dataset are in their own directory. Each person using the social network has an id which is the name of the data file (e.g., 1209.txt has the id of 1209). Each file contains:

id ← stored as a string even though it's a number
name ← could be a single word or a full name with spaces
hobbies ← list of hobbies, each one separated with a comma
friendId ← a list of their friends, listing their id only, and one per line
friendId
…

There is also an additional file in each dataset that has a list of all the users' ids. This will be used to read in all nodes (vertices) in the graph even if there are some that are not connected. This can be named anything, as you will send in the name of this file to command-line arguments (therefore do not hard-code in the name of the file!)

Part 1

This part of the program creates a graph data structure linking friends together based on a social network. This is very similar to your previous homework, but this time the nodes are *not* directed; therefore, when a friend is listed as a friend then you need to add an edge from each vertex to the other. For example, if Sally lists me as a friend in her file but I do not list her id as a friend in my file – that doesn't matter → we are friends and you will need an edge between both of our vertices to each other.

You will also need to set up which network group they are in (e.g., district ID from HW 8).

Part 2

The second part is to work with the hobbies. Store these in a hashtable as an array of pointers to the vertices. Use the simple hashing method of simply adding 1 to the key for collisions (and no chaining!). You will print out how many collisions occur. Therefore your number of collisions should match our answer keys (we are using the same hashing function and hash table size) and your output when you loop through the hash table to display the hobbies should be in the same order as well.

Be sure this works on the large dataset! There is something extra you need to remember to do!

Output

Example run of your program using Dataset2:

```
./a.out Dataset2 Dataset2/userids.txt
```

Expected output:

```
Creating graph of social network ids from file: Dataset2/ids.txt
Processing dataset: Dataset2 (7 files)
=======================================
 NUMBER OF COLLISIONS ON ADDING HOBBIES: 3
=======================================
GRAPH CONNECTIONS ====================
1: [1209] Eliza-->Jaime***Zhrima***Zoe
2: [1999] Chazu-->Kaira
1: [1988] Jaime-->Eliza***Zuhaima***Zhrima
1: [2299] Zuhaima-->Jaime
1: [2440] Zhrima-->Eliza***Jaime***Zoe
2: [2229] Kaira-->Chazu
1: [3728] Zoe-->Eliza***Zhrima
=======================================
DISPLAYING HOBBY INTERESTS ============
golf: Jaime, Zuhaima, Zhrima
reading: Chazu, Zuhaima, Kaira, Zoe
frisbee golf: Eliza, Zuhaima
mountain biking: Eliza, Zoe
skiing: Chazu, Zoe
programming: Eliza
travel: Eliza, Zhrima
trekking: Zhrima
trolling: Zoe
painting: Kaira, Zoe
```

Note – your code also needs to work on the large dataset Dataset3! Use the discussion board to figure out from your peers what the output should be.
To run on Dataset 3, use: ./a.out Dataset3 Dataset3/ids.txt

Submitting Your Code:

Submit your assignment named **Graph.cpp** and **Driver.cpp** as a zipped file named **HW9XC_[Firstname]_[Lastname].zip** to Moodle.