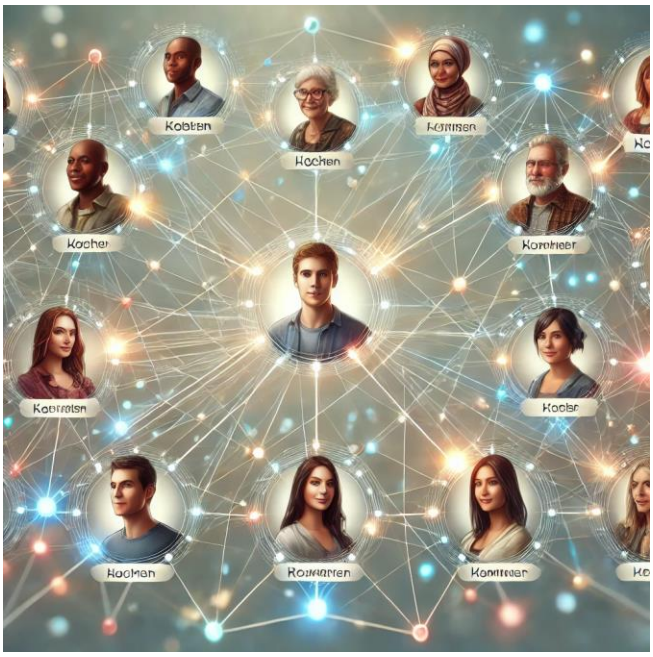
PROGRAM 4 ASSIGNMENT INSTRUCTIONS

CSC 1310-001, SPRING 2025

Degrees of Separation in Social Media Networks



DESCRIPTION

Degrees of separation theory plays a role in how social media platforms make friend recommendations. Social networks model users and their connections as a graph where nodes are users and edges are friendships or interactions.

You will write a C++ program to read and analyze a social network graph using an adjacency matrix. You will use the Breadth-First Search (BFS) algorithm to determine the degree of separation (distance) between each person and all others in the graph.

CONCEPTS

- Graphs
- Breadth-First Search (BFS) algorithm (contains a Queue) - https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
- Hash Tables (C++ unordered_map library)
- Six Degrees of Separation (https://en.wikipedia.org/wiki/Six_degrees_of_separation)

DUE DATE

Tuesday, April 22, 2025
You may submit up to 2 days late with 10 points off per day late.

YOU ARE GIVEN

graph1.txt — This file contains the number of people in the social network and all the relationships (edges) between them.

You can find this file in the Program 4 Assignment in ilearn.

IMPORTANT RULES YOU MUST FOLLOW TO AVOID A ZERO

As stated in the syllabus, you may not use any of the following in your code:

- while(true) or while(1)
- Goto statements
- Recursively calling the main function
- exit()
- Ternary operators (any syntax requiring a '?')

GENERATIVE AI / ACADEMIC INTEGRITY POLICY FOR PROGRAM 4

- You may only use generative AI to help you with troubleshooting your code. If you do use AI, you must submit a copy of your entire AI conversation as part of your submission.
- You may not:
  - Copy/paste any part of this assignment or the sample output into AI tools
  - Ask AI to create functions or write code for you
- **Share more than 1–2 lines of code with anyone other than a TA or your instructor**

If you receive help or find code ideas online, cite your source in a comment and ensure you understand the code completely.

PROGRAM REQUIREMENTS AND FLOW

1. **Graph Representation**: create a class GraphMatrix (in a file named **GraphMatrix.h**) that uses a dynamically allocated 2D array to represent an adjacency matrix.

2. **Reading Data**: You will read the data from a text file that the user enters in (you were given a sample – graph1.txt). The first number is the number of vertices and the remainder are the vertices that are adjacent. After reading in the # of vertices & storing it, then read in the vertices that are adjacent and add an edge to the graph. Then use **unordered_map<int, string>** ~~and unordered_map<string, int>~~ to map between vertex IDs and person names. Note: the original assignment said to have two undordered_maps but you really only need one – to map vertex ID to name.

3. **Print** the **graph's adjacency matrix** by calling the printGraph() function for the GraphMatrix object.

4. **Breadth-First Search (BFS):** Implement the BFS algorithm via a helper function in your source file to determine the shortest path from one node (person) to all others. You may use the algorithm from the geeks for geeks website: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/. Store distances in a vector (which should be returned from the BFS function) and **print the result as a matrix** as you see in the **sample output**. Also, keep track of the person with the largest number of 1 degree of separation – this is the most well-known person.

5. **Print** the most well-known person.

CREATE ONE GRAPH TEXT FILE

You can use the provided graph1.txt to test your program against the sample output provided, but you should also create your own graph.

Name the text file **graph2.txt** and it can contain network data for your friends/family, made-up characters, or characters from your favorite TV show or movie. Please have it contain at least five vertices and six edges.

SUBMISSION INSTRUCTIONS

Submit the following files in a zipped folder on iLearn:

- **yourusername_prog4.cpp** — contains your main function and bfs function
- **GraphMatrix.h** — your adjacency matrix class
- **graph1.txt** – the one given to you
- **graph2.txt** – the one that you create
- Document showing your AI conversations (if applicable)

A LITTLE HELP ON PRINTING THE DEGREE OF SEPARATION MATRIX & BFS ALGORITHM

Note: this whole section is new and was not in the original assignment document and is only provided to help with clarification on how to do this part of the assignment. For the degree of separation matrix, you can either update the actual graph matrix with the distances vector for each vertex and then print it. Or the way that I did it is to print the vectors line by line from the main function. I am going to put pseudocode below of how I did this in the main function and the bfs function.

MAIN FUNCTION PSEUDOCODE FOR PRINTING "DEGREE OF SEPARATION MATRIX"

```
Print "Degree of Separation Matrix" header

For each person (i) in the graph:
    Print the person's name as a row label

    Call the BFS algorithm using person i as the starting point
    Store the result in a list of distances

    Initialize a counter to count how many people are directly connected (degree 1)

    For each distance in the list:
        If the distance is -1, print a dot (.) to show no connection
        Otherwise, print the number (degree of separation)
        If the distance is 1, increase the counter

    After the row is printed:
        If this person has more direct (1st-degree) connections than any seen so far:
            Store this number and this person's name

After all rows are printed:
    Print the person who has the most direct (1st-degree) connections
```

BFS FUNCTION PSEUDOCODE

```
Function BFS(graph, start_person):
    Create a vector called 'distance' to keep track of how far each person is from the start
        - Initialize all distances to -1 (not visited)
    Set the distance for the start person to 0
    Create a queue and add the starting person to it

    While the queue is not empty:
        Take the first person out of the queue (store this person in a variable – I called it curr)
        For every person (i) in the graph:
            If there is a connection from the current person (curr) to that person (i)
               AND that person (i) hasn't been visited yet (distance is -1):
                Set their distance to be the current person's distance + 1
                Add them to the queue

    Return the vector of distances
```

GRAPH1.TXT

graph1.txt

File Edit View

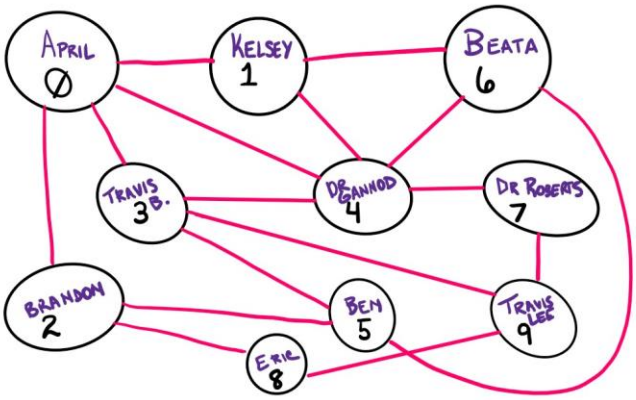
```
10
0/April Crockett,1/Kelsey Rainey
0/April Crockett,2/Brandon Vandergriff
0/April Crockett,3/Travis Brummett
0/April Crockett,4/Dr. Gannod
1/Kelsey Rainey,4/Dr. Gannod
1/Kelsey Rainey,6/Beata Kubiak
2/Brandon Vandergriff,5/Ben Burchfield
2/Brandon Vandergriff,8/Eric Brown
3/Travis Brummett,5/Ben Burchfield
3/Travis Brummett,9/Travis Lee
3/Travis Brummett,4/Dr. Gannod
4/Dr. Gannod,6/Beata Kubiak
4/Dr. Gannod,7/Dr. Roberts
5/Ben Burchfield,6/Beata Kubiak
7/Dr. Roberts,9/Travis Lee
8/Eric Brown,9/Travis Lee
```

The original assignment had a mistake on this line.
This is the corrected version. It should be 6/Beata Kubiak instead of 6/Dr. Roberts

Note: This did not affect the sample output solution at all since the vertex number was correct and because the map from 6 to Beata Kubiak happens after this line and so it gets overwritten. I just didn't want it to confuse anyone.

DRAWING OF GRAPH1:

(added on 4/14/2025) to help with clarification



SAMPLE OUTPUT

User input is highlighted in yellow.

```
What is the name of your text file that has your graph data (filename.txt)?
File: graph1.txt

Resulting Graph Adjacency Matrix:

     0  1  2  3  4  5  6  7  8  9
   ---------------------------------------
 0|   .  1  1  1  1  .  .  .  .  .
 1|   1  .  .  .  1  .  1  .  .  .
 2|   1  .  .  .  .  1  .  .  1  .
 3|   1  .  .  .  1  1  .  .  .  1
 4|   1  1  .  1  .  .  1  1  .  .
 5|   .  .  1  1  .  .  1  .  .  .
 6|   .  1  .  .  1  1  .  .  .  .
 7|   .  .  .  .  1  .  .  .  .  1
 8|   .  .  1  .  .  .  .  .  .  1
 9|   .  .  .  1  .  .  .  1  1  .
```

Degree of Separation Matrix:

	0	1	2	3	4	5	6	7	8	9
April Crockett	0	1	1	1	1	2	2	2	2	2
Kelsey Rainey	1	0	2	2	1	2	1	2	3	3
Brandon Vandergriff	1	2	0	2	2	1	2	3	1	2
Travis Brummett	1	2	2	0	1	1	2	2	2	1
Dr. Gannod	1	1	2	1	0	2	1	1	3	2
Ben Burchfield	2	2	1	1	2	0	1	3	2	2
Beata Kubiak	2	1	2	2	1	1	0	2	3	3
Dr. Roberts	2	2	3	2	1	3	2	0	2	1
Eric Brown	2	3	1	2	3	2	3	2	0	1
Travis Lee	2	3	2	1	2	2	3	1	1	0

The most well-known person with the fewest degrees of separation from others is Dr. Gannod with 5 1st degree connections in the social graph provided.