CCDSALG Term 3, AY 2024 – 2025

Project 2 Documentation – Social Network (Undirected Graph Application)

DECLARATION OF INTELLECTUAL HONESTY / ORIGINAL WORK

I declare that the project that I'm submitting (as part of a group) is the product of my own intellectual efforts. No part of the project was copied from any source, and no part was shared with another person.

NAME#1: CALUPIG, EVAN RILEY L. SIGNATURE#1: [illegible] SECTION: S11

The following are my specific contributions to MCO2:

1. Created the initial directory structure (src, include, bin, etc.)
2. Planned the separation of problems into different modules.
3. Wrote initial makefile.
4. Wrote the functions to generate degree and set(Outputs 1 and 2.)
5. Implemented DFS, BFS, and the queue data structure.
6. Wrote the initial testing script to automate running the program.
7. Added function documentation and code styling.

Based on my honest estimate, and in consultation with my groupmates, I contributed approximately **50%** to the entire project.

Example: if you encoded 40% it would mean you did 40% of the entire project on your own from say from conceptualization, to testing and documentation, while the remaining 60% were done by your groupmates. Your contribution PLUS the contribution of your groupmates should total to 100%.

NAME#2: DONGUINES, JON CHESTER C. SIGNATURE#1: [illegible] SECTION: S11

The following are my specific contributions to MCO2:

1. Implemented graph data structure.
2. Implemented file input and parser.
3. Implemented string sorting module.
4. Implemented list and matrix output functions (Outputs 3 and 4).
5. Created and tested three test case files.
6. Wrote function documentation.

Based on my honest estimate, and in consultation with my groupmates, I contributed approximately **50%** to the entire project.

1. Indicate how to compile (if it is a compiled language) your codes, and how RUN (execute) your program from the COMMAND LINE. Examples are shown below highlighted in yellow. Replace them accordingly. Make sure that all your group members test what you typed below because I will follow/copy-paste them verbatim. I will initially test your solution using some of the sample input text file that you submitted. Thereafter, I will run it again using my own test data:

- How to compile from the command line

```
CCDSALG> gcc -Wall -c app.c file_io.c graph.c main.c queue.c set.c sort.c traversals.c
CCDSALG> gcc -Wall app.o file_io.o graph.o main.o queue.o set.o sort.o traversals.o -o main
```

Next, answer the following questions:

a. Is there a compilation (syntax error) in your codes? (YES or NO) **NO.**
   WARNING: the project will automatically be graded with a score of **0** if there is syntax error in any of the submitted source code files. Please make sure that your submission does not have a syntax error.

b. Is there any compilation warning in your codes? (YES or NO) **NO.**
   WARNING: there will be a 1 point deduction for every unique compiler warning. Please make sure that your submission does not have a compiler warning.

- How to run from the command line

```
CCDSALG> main
```

Did you do the BONUS part (YES or NO) **NO**.

NOTE: Please do NOT submit a solution if it is not working correctly based on the exhaustive testing that you have done.

If you did the BONUS part, indicate below:

- How to compile from the command line

- How to run from the command line

2. How did you implement your Graph data structure? Did you implement it using adjacency matrix? adjacency list? Why? Explain briefly (at most 5 sentences).

The graph data structure was implemented using an adjacency matrix representation. The GraphType struct uses a 2D array as an adjacency matrix to represent edges. It also stores a list of vertex IDs, the number of vertices, a list of neighbors, and the number of neighbors per vertex. This was chosen since the number of vertices was low and already defined. There was also no need to modify the graph structure.

3. How did you implement the DFS and BFS traversals? What other data structures (aside from the graph representation) did you use? Explain your algorithm succinctly.

a. DFS Traversal:

Our Depth-First Search (DFS) was implemented using a recursive approach. The core logic resides in a helper function that is called recursively to explore the graph's depth.

Data Structures:

- visited[] - int array that acts as flag to track which vertices have been visited
- result[] - string array to store the final sequence of visited vertices
- neighbours[] - string array to ensure traversal order

Algorithm:

1. Visit starting vertex, mark it as visited in visited[], add its name to result[]
2. Store neighbours in neighbours[]
3. Sort neighbours[] alphabetically
4. Iterate through the now sorted neighbours[] array. If that neighbour has not been visited yet, call DFS

b. BFS Traversal:

Our Breadth-First Search (BFS) was implemented iteratively to explore the graph level by level.

Data Structures:

- queue - stores the stores integer indices of vertices to be visited
- visited[] - int array that acts as flag to track which vertices have been visited
- result[] - string array to store the final sequence of visited vertices
- neighbours[] - string array to ensure traversal order

Algorithm:

1. Visit starting vertex, mark it as visited in visited[], enqueue.
2. While queue is not empty, run the loop:
   a. dequeue, add dequeued vertex name to result[]
   b. Store neighbours of dequeued vertex in neighbours[]
   c. Sort neighbours[] alphabetically
   d. Iterate through sorted neighbours. If that neighbour has not been visited yet, mark as visited, and enqueue.

4. Disclose **IN DETAIL** what is/are NOT working correctly in your solution. Please be honest about this. NON-DISCLOSURE will result in severe point deduction. Explain briefly the reason why your group could not make it work.

The following are NOT working (buggy):

a. None

We were not able to make them work because:

a. None

5. What do you think is the level of difficulty of the project (was it easy, medium or hard)? Which part is hard (if you answer was "hard") and why?

Evan Riley L. Calupig: Medium. Once you analyze the problem, the solution is pretty much straightforward. The more difficult parts of the problem for me were the more menial tasks, like the file IO and parsing.

Jon Chester C. Donguines: Medium. At first, it was a little tricky to integrate the graph structure with some of the functions (traversals, set representation).

6. Fill out the table below.

REQUIREMENT	**AVE. OF SELF-ASSESSMENT**
1. Correctly produced Output Text File #1 (Set of Vertices & Edges)	**10** (max. 10 points)
2. Correctly produced Output Text File #2 (Vertices With Degrees)	**10** (max. 10 points)
3. Correctly produced Output Text File #3 (Adj. Matrix Visualization)	**10** (max. 10 points)
4. Correctly produced Output Text File #4 (Adj. List Visualization)	**10** (max. 10 points)
5. Correctly produced Output Text File #5 (BFS Traversal)	**22.5** (max. 22.5 points)
6. Correctly produced Output Text File #6 (DFS Traversal)	**22.5** (max. 22.5 points)
7. Documentation	**10** (max. 10 points)
8. Compliance: deduct 1 point for every instruction not complied with (examples: incorrect output file names, extraneous contents in output files, etc.)	**5** (max. 5 points)
9. Bonus :-)	**0** (max. 10 points
TOTAL SCORE	**100** over 100.

NOTE: The evaluation that the instructor will give is not necessarily going to be the same as what you indicated above. The self-assessment serves primarily as a guide on how your project will be evaluated.

***** THE END *****