CIS\*2520 — Assignment #4

Fall 2024

**Due**: Friday, November 29, 2024 @ 23:59

Please submit your assignment solutions as **one zip file** (named as YOUR/UoG/ID\_a4.zip, e.g. 1234567\_a4.zip) to Dropbox under Assignment 4 before the due date.

You are granted a penalty-free grace period for 48-hours. The grace period ends on Sunday, December 1 23:59. After this time, you cannot submit the assignment. The late assignment (no submission after the grace period) will be marked as ZERO.

**Please note: No further extension can be requested on this assignment. You must submit your assignment 4 by Sunday, December 1 23:59.**

Please refer to the Course Outline and Academic Integrity Video to ensure you understand and comply with the University’s Academic Integrity Standards.

**Graph**

You will write a C program to process a graph using its adjacency matrix stored in a .txt file as the input (a sample file is provided). The program should read the graph and provide several functionalities for interacting with the graph.

**Input:**

The adjacency matrix (denoted as A) for the graph will be stored in a .txt file, where rows and columns represent vertices (numbered 1, 2, 3, …), and the values in the matrix represent the edge weights. If , it means that there is no edge between the vertex *i* and *j*.

**Requirements:**

1. **Command-line argument:**
   * The program should accept the filename of the .txt file containing the graph's adjacency matrix as a command-line argument.
   * Example usage: ./a4 example-graph.txt (the executable is named a4).
2. **Menu of Options:** Upon running the program, it should present the user with a menu of options. The user should be able to type the number of an option, and the program should execute the corresponding functionality.
   * + **Display Adjacency List**
     + **Perform Breadth-First Search (BFS)**
     + **Perform Depth-First Search (DFS)**
     + **Find Shortest Path using Dijkstra's Algorithm**
     + **Exit**
3. **Option Details:**
   * **Option 1: Display Adjacency List**

When this option is selected, the program should print the adjacency list representation of the graph. Each vertex should list its connected vertices along with weights of the edges.

* + **Option 2: Perform Breadth-First Search (BFS)**

When this option is selected, the program should perform a Breadth-First Search (BFS) traversal **starting from vertex 1**. The program should print the vertices in the order they are visited.

* + **Option 3: Perform Depth-First Search (DFS)**

When this option is selected, the program should perform a Depth-First Search (DFS) traversal **starting from vertex 1**. The program should print the vertices in the order they are visited.

* + **Option 4: Find Shortest Path (Dijkstra's Algorithm)**

When this option is selected, the program should compute and print the shortest paths from **vertex 1** to all other vertices using Dijkstra's algorithm.

* + **Option 5: Exit**

When this option is selected, the program should exit.

# **Guidelines**

* Ensure your program reads the adjacency matrix from the file correctly.
* If the input file is missing or cannot be opened, the program should display an appropriate error message.
* After each operation, the program should return to the main menu until the user chooses to exit the program.
* You should use the **linked list** to store the graph’s Adjacency List and perform the required operations.
* Additional guideline is in **CIS2520\_F24\_A4.md**. Please ensure you read through this document.
* You must use the template files provided for this assignment. You **cannot** change the names of the template files, and the names of the functions in the template files.
* You must create a **readme** file. The information in readme file can be the same as in Assignment 2. You are welcome to use the previous template and add more info to it as you want, such as the testing input/output from the samples given to you.
* You must create a **makefile** (one single makefile) for compiling the program. The makefile should be able to generate an executable file **named** a4.
* Yu must compile with “gcc -Wall -std=c99 -pedantic”. **The code must be tested on the school server before submission**.

**Submission requirement**:

* Only the following files are required and should be submitted. All the files should be under the folder named YOUR/UoG/ID\_a4.
  + *main.c, graph.h*, and *graph\_functions.c*
  + *README*
  + *makefile*
* No other files should be included in your submission, such as .o files, txt files, etc..
* Once you have everything, zip the folder and submit it via **CourseLink Dropbox**.

# **Information on Grading**

* The grading of this assignment is heavily based on **Input/Output test**. Therefore, you must use the given sample output and **ensure your output matches with the sample out.**
* There are points allocated for style, comments, documentation, readme, and makefile. For example, in each of the C files, any function should have a brief comment describing its purpose. Also, any section of code where it is not easily apparent what the code does should have a short comment. Don’t forget indentation.
* To ensure you receive full marks, please fill out all functions in the given function templates.
* Any compilation error or warning will result in a mark deduction appropriate to the severity of the error. Memory leaks and memory errors will also result in mark deductions.
* If your submission fails to compile or execute, you may receive **ZERO** for this assignment.
* If your submission does not contain a Makefile or a working Makefile, you may receive **ZERO** for this assignment.