Homework 4 – Programming Project

EECE 4040 – Fall 2020

Due Wednesday, October 28 before midnight

*The **leader** of each group is to upload a .cpp file with the **source code** for your* C++ *program as well as a file with **output for a sample run**. For ease of grading, all the C++ code for your program should be included in a **single** file and designed using the C++ Visual Studio Platform. It should be well-commented and the output user-friendly.*

Topics covered: **Implementing a digraph, topological sort of DAGS.**

Purpose: **The purpose of this programming project is to gain experience with implementing digraphs and topological sort.**

This program involves writing a C++ program for implementing an ADT Directed Graph and performing a topological sort of a DAG. You are to implement the directed graph using pointer-based adjacency lists (using an array of header nodes and a linked list for each header node). Your class `Digraph` should include constructors and a destructor, operations of edge addition, edge deletion, etc., as well as the operation of topological sorting and acyclic check. You are to implement the topological sorting operation **using DFT** as discussed in the lecture video and Chapter 5 of the textbook.

Using appropriate, user-friendly prompts have the user input a set of tasks into an array of strings, e.g.,

1. Paint walls
2. Install electrical wiring
3. Lay foundation
4. Do roofing
5. Put up drywall
6. Install plumbing
7. Frame house

etc.

After the user has entered the tasks, your program should then have the user specify (using user-friendly prompts) an order relation on pairs of tasks, e.g.,

3 1 (indicates that Task 3 must precede Task 1)
7 5 (indicates that Task 7 must precede Task 5)
5 1 (indicates that Task 5 must precede Task 1)
etc.

Your program then applies the topological sort operation using DFT discussed in the lecture video. Also see textbook 5.5 (Chapter 5, Section 5), Pages 231-234. In the

pseudocode on Page 232 an array is used to store the topological-sort list. In this program, I would like you **to use a linked list** instead.

In the case where the digraph is not acyclic, your program should output the error message that the directed graph is not a DAG. The latter can be achieved by assigning each node of the DAG its topological-sort label, i.e., the order it occurs in the topological-sort list. If the label on the tail of any edge is greater than the label on its head, then the digraph is not a DAG; otherwise it is.