CSCE 221 Cover Page
Homework Assignment

First Name: Thu Last Name: Nguyen UIN: 230002254

User Name: tqn14 E-mail address: tqn14@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources	Lectures	Online	Textbook		
People	Dr. Teresa Leyk	various sources	Mark A. Weiss		
Web pages (provide URL)	eCampus	listed below	printed		
Printed material					
Other Sources					

https://www.youtube.com/watch?v=_LuIvEi_kZk
http://www.cplusplus.com/reference/vector/vector/begin/
https://docs.oracle.com/cd/E19504-01/802-5880/6i9k05dhg/index.html
https://stackoverflow.com/questions/15099707/how-to-get-position-of-a-certain-element-in-strings-vector-to-use-it-as-an-inde
https://stackoverflow.com/questions/35050065/c-error-no-match-for-operator-operand-types-are-vehicle-and-const-ve
http://www.cplusplus.com/doc/tutorial/exceptions/
http://www.cplusplus.com/reference/algorithm/find/

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Thu Nguyen Date 06/27/2020

The Programming Assignment Report Instructions
CSCE 221

1. The description of an assignment problem.

   The assignment consists of 3 parts. Part 1 concerns the implementation of a graph data structure. Part 2 is about the implementation of topological ordering for a Directed Acyclic Graph. Data of each graph is given in a text file as adjacency list, and a topologically sorted result should be returned after taking in the input and performing the sorting.

2. The description of data structures and algorithms used to solve the problem.

   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)

   The assignment requires the use of graph data structure, which has a Vertex structure with the label member to return the integer values of the vertices, the indegree member to store the indegree of each vertex, the top_num member to store the topological order of each vertex.

   Each vertex is store in a vector of Vertex type. The adjacency nodes of a vertex are stored in a linked list at the position matching its position in the vector.

   (b) Write about the ADTs implementation in C++.

   The functions implemented are buildGraph, displayGraph, topological Sort, compute indegree and print top sort. buildGraph reads input data into a vector and linked list of each type. Display Graph shows the input in the form of adjacency list. Topological Sort uses the algorithm to sort the data topologically. First compute the indegree from the adjacency list by counting the appearance of each vertex in the adjacency list. Then push in the vertext that has 0 indegree into a queue, decrement the indegrees of their adjacent nodes, then extract it from the queue. Only push the nodes with 0 indegree into the queue. Repeat the same process with other nodes in the queue until it's empty.

   The assumption about input data: graph nodes are numbered consecutively from 1 and there is no gap in the node numbering.

   We can use a stack instead of a queue. As long as the push and pop functions are done in constant time so that we can sort the nodes efficiently. The algorithm uses this type of data structure because inserting and removing takes $O(1)$, which is fast.

   The algorithm can detect cycles in the graph because it keeps decrementing the indegree of the vertices. When there is a cycle, the nodes' indegree in that cycle won't be decremented as well as their adjacent nodes, thus they won't be added to the queue, therefore the count in topological sort function won't be updated. In the end, if we check for count and the number of vertices, they are not the same, which means there are cycles in the graph.

   (c) Analyze the algorithms according to assignment requirements.

   For computing the indegree, even though there are nested loops, the cost is $O(|E|+|V|)$ since they are executed at most $|E|+|V|$ times for a strongly connected graph.

   Similarly, the cost of topological sort function is $O(|E|+|V|)$ because adding the inserting to the queue take $O(1)$,and going through each edge for at most $|E|+|V|$times to decrement the indegree.

   Printing the topological sort takes $O(|V|)$ since going through V vertices and take $O(1)$ to find each topological number.

   BuildGraph takes $O(|E|+|V|)$ because there are V vertices and E edges.

3. A user guide description how to navigate your program with the instructions how to:

(a) compile the program: specify the directory and file names, etc.
run and combine all the files into 1 executable `myprogram` by `g++ -std=c++17 graph.cpp topological_sort main.cpp -o myprogram`

(b) run the program: specify the name of an executable file.
`./myprogram input.data`

4. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

   input.data for the example in the assignment prompt, input2.data for CSCE course scheduling, input3.data for LA language course scheduling, input-cycle.data for a cyclic graph.

5. Provide types of exceptions and their purpose in your program.

   (a) logical exceptions (such as deletion of an item from an empty container, etc.).

   Exceptions thrown for out of bound (used to find the index of each vertex in vector node_list) and cycles detected in the graph.

   (b) runtime exception (such as division by 0, etc.)

6. Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.

```
tqn14@LAPTOP-PEC7250E:/mnt/c/Users/quynh/Documents/csce221/pa3$ g++ -std=c++17 graph.cpp topological_sort.cpp main.cpp -o myprogram
tqn14@LAPTOP-PEC7250E:/mnt/c/Users/quynh/Documents/csce221/pa3$ ./myprogram input.data
1:2 4 5
2:3 4 7
3:4
4:6 7
5:
6:5
7:6
1 2 3 4 7 6 5
tqn14@LAPTOP-PEC7250E:/mnt/c/Users/quynh/Documents/csce221/pa3$ ./myprogram input2.data
1:3
2:3 8
3:4 5 6 8
4:7
5:7
6:
7:
8:
1 2 3 4 5 6 8 7
tqn14@LAPTOP-PEC7250E:/mnt/c/Users/quynh/Documents/csce221/pa3$ ./myprogram input3.data
1:2 4
2:5 7 8
3:6 8
4:5
5:6 9
6:
7:
8:
9:
1 3 2 4 7 8 5 6 9
tqn14@LAPTOP-PEC7250E:/mnt/c/Users/quynh/Documents/csce221/pa3$ ./myprogram input-cycle.data
1:2 4 5
2:3 4 7
3:4
4:6 7
5:4
6:5
7:6
terminate called after throwing an instance of 'cycleGraph'
  what():  there are cycles in the graph
Aborted (core dumped)
tqn14@LAPTOP-PEC7250E:/mnt/c/Users/quynh/Documents/csce221/pa3$
```