CSE 310 Assignment #7
(Max. Points: 30)

Due on: Monday, Nov. 27, 2017, 11:59pm Arizona time

General Instructions:

- This is an individual assignment, please do not collaborate. *Make sure that you write every line of your own code. Using code written by someone else will be considered a violation of the academic integrity and will result in a report sent to the Dean's office.*
- It must be submitted online through Blackboard submission link! We don't accept submissions through emails.
- Submission link will be closed automatically once the due date/time is past and **no late assignment will be accepted**.
- You will be allowed 3 times to submit the assignment, but we will only grade your last submission.

Objectives

- Practice depth-first-search on a directed graph represented by an adjacency-list.

Assignment Description

1. You need to implement a graph data structure using an adjacency list to keep track of which university football team beat which team. Thus the nodes of your graph will be university names, and each edge, *AUniversity -> BUniversity*, represents that *AUniversity* football team beat *BUniversity* in some games. Also, university names need to be stored in alphabetical order, *i.e.*, the node names that represent "*from*" nodes should be in alphabetical order, and also "*to*" nodes from each "*from*" nodes should be stored in each linked list in alphabetical order.

2. Your program needs to read from two files, the first file contains all the commands the driver's program will execute and the second file contains the graph's information. After compiling your program using *makefile*, it should generate an executable file named "***A7p1***". For example, it will be executed using the following command:

```
./A7p1 commands1.txt edges1.txt
```

For testing simplicity, always assume the first file's name is *commands1.txt* and the second's file name is *edges1.txt*. ***commands1.txt*** contains a list of commands and ***edges1.txt*** contains the information on edges of the graph.

3. Your driver's program will only takes the following commands:

- *graph*
- *end_graph*
- *print_graph*
- *depth_first_search*
- *quit*

The program should print out each entered command before processing it. Below please find the description for each command.

graph,startLine,endLine

A real example of this command can be:

```
graph,0,4
```

When your driver program sees this command, it should start reading from ***edges1.txt*** which contains the graph edges information. It should read from the line specified by the first number after "*graph*", and read up to the line specified by the second number, and populate its data into your graph, and create an adjacency list representation of the graph.

For above example, the driver program should read from line 0 to 4 of ***edges1.txt***, which means that it should read 5 lines (5 edges), and create a graph from them. In case the second number is larger than the last line of the file, the program should just read till the end of the file.

File ***edges1.txt*** contains the information on edges in the following format:

```
Arizona State,34,New Mexico,10
Boise State,52,Idaho State,24
Florida State,14,Boston College,7
Akron,52,Savannah State,9
Arizona,77,Northern Arizona,13
Arkansas State,70,Missouri State,7
......
```

Each line contains a result of a college football game where:

- the first string is the university that won the game,
- the second string is the winner's score,
- the third string is the university that lost the game,
- the forth string is the loser's score.

Above four strings are separated by commas.

For this assignment, you may consider using an array of linked list to store the graph information. The university names need to be stored in alphabetical order, both for the "from" nodes in the array and also "to" nodes in each linked list. Your program needs to read edges from ***edges1.txt*** within the specified lines to determine all university names that are used as either a "from" node or a "to" node, and create an array of linked lists with the size being the number of such universities.

end_graph
This command should free the memory allocated for your graph, *i.e.*, free memory of the adjacency list

print_graph

This command should print the content of the graph, using the following format:

```
Arizona State beat: Florida State(43-14), New Mexico(58-23),
Oregon(12-9)
Southern Arizona beat: none
Texas A&M beat: New Mexico(32-12)
UCLA beat: Arizona State(62-27)
```

"Arizona State beat: Florida State(43-14), New Mexico(58-23), Oregon(12-9)" means that:

- There is an edge from "*Arizona State*" to "*Florida State*", an edge from "*Arizona State*" to "*New Mexico*", and an edge from "*Arizona State*" to "*Oregon*".
- The numbers within parentheses indicates their scores for each game where the first number is the winning score and the second number is the losing score.

From every university that is used within the specified lines of the ***edges1.txt*** needs to be listed here, and to which university that it has an edge. Please print "*none*" if there is no edge coming out of some university (see above example).

depth_first_search

For this command, starting from the node (the university) that comes first in alphabetical order, it should perform a Depth First Search on the graph and prints out:

- the ***pi*** array values (DFS .π attribute for each node)
- starting time array values (DFS .*d* attribute for each node)
- finishing time array values (DFS .*f* attribute for each node)

It should use the following format:

```
The array pi content:

pi[Arizona State] = none
pi[Arkansas] = Florida State
pi[Florida State] = Arizona State
pi[Mississippi State] = none
pi[New Mexico] = Arizona State
pi[Oregon] = Arizona State
pi[Texas A&M] = Mississippi State
pi[UCLA] = Oregon

The array discoverTime content:

discoverTime[Arizona State] = number 1
discoverTime[Arkansas] = number 3
discoverTime[Florida State] = number 2
discoverTime[Mississippi State] = number 13
discoverTime[New Mexico] = number 6
```

```
discoverTime[Oregon] = number 8
discoverTime[Texas A&M] = number 14
discoverTime[UCLA] = number 9

The array finishTime content:

finishTime[Arizona State] = number 12
finishTime[Arkansas] = number 4
finishTime[Florida State] = number 5
finishTime[Mississippi State] = number 16
finishTime[New Mexico] = number 7
finishTime[Oregon] = number 11
finishTime[Texas A&M] = number 15
finishTime[UCLA] = number 10
```

quit

This command quits the program and exit.

Please see the sample input files, *commands1.txt*, *edges1.txt*, and sample output file *output1.txt*. Note that all outputs should be printed to a console instead of writing to a file.

Implementation/Documentation Requirements

- You need implement this program using C++ and it has to read from two files (argv[1] and argv[2]) .
- Your program needs to compile and execute in *general.asu.edu*.
- You need to define *graphInsertNode*, *graphInsertEdge*, *depthFirstSearch*, and *destructor* to free the memory for your graph.
- You must use the provided data sets.
- You are not allowed to use any predefined data structures (such as ones in the library in C++, etc.) except arrays and strings, you need to build your own data structures and operations associated with them (such as insert or search).

Copy any codes from other people's programs is considered to be cheating and will lead to a report to the Dean and you will be penalized. Also check the rules stated in the syllabus of the course regarding the academic integrity.

- At the top of each source code file, add the following information:

```
// Name of Author(s):
// ASU ID:
// Homework Number:
// Description:
```

A sample test case provided

commands1.txt edges1.txt output1.txt

What to turn in

Under the submission link on Blackboard, submit a zip file named ***youFirstNameyourLastName.zip*** containing the following:

1. One or more files of your well documented C++ source code (with a file extension *.cpp* or *.h*) implementing the commands required.

2. A Makefile that compiles your program to an executable named "***A7p1***" on the Linux machine *general.asu.edu*. Our TA will write a script to compile and run all student submissions on *general.asu.edu*; therefore executing the make command must produce the executable file "***A7p1***" in the same folder as your source code files. After such ***A7p1*** file is produced, then we should be able to execute your program using the command:

```
./A7p1 command1.txt edges1.txt
```

where *command1.txt* and *edges1.txt* are file names, but we should be able to use files with a different name with a content in the format specified in the project statement..

Your program must read from a keyboard, not a file.

Error Handling

Your program will be tested with other input besides the sample input given, thus is expected to be robust.

Grading Rubric

- 2 pts – Documentation (correct and complete code description and comments, header for each function/method, each file should have your name at its top) and good indentation and spacing (easy to read)
- 1 pt – makefile is submitted and it works correctly to generate A7p1 executable file
- 12 pts – Each of the following functions are defined correctly: *graphInsertNode, graphInsertEdge, depthFirstSearch, and destructor.*
- 14 pts – Correct output for the test case(s)

Total points: 30