CS575: Programming Assignment 3
Due at 11:59:59PM, March 30, 2020

1. [45%] Implement the longest common subsequence (LCS) algorithm using the dynamic programming method that was discussed in class. (No credit will be given if you implement a brute force algorithm, which does exhaustive comparisons between two input strings, or any other algorithm unless you prove your algorithm is correct and more efficient than the LCS algorithm described in Chapter 7.) Save your source code in a file and name the file as *yourlastname_yourfirstname_pa3_lcs.cpp*.

   Make sure that your program can take **any** two input strings in the Linux command line and print the LCS found between the two input strings. (Assume that a string consists of at most 100 alphabetic characters.) For example, student Joe Smith types "Smith_Joe_pa3_lcs abc afgbhcd" in the command line to find the LCS between string "abc" and string "afgbhcd". Again, your program should work for arbitrary two input strings. No credit will be given, if your program only works for some specific strings, but fails to find the LCS for other strings.

2. [45%] Randomly create an undirected complete graph as follows:
   - Create n vertices where n is randomly selected between 5 and 10. Display the selected n value.
   - Create an n x n adjacency matrix A. Randomly assign a weight to each edge (i,j) where $1 \le i, j \le n$. Specifically, make A[i,j] = 0 if i = j. If $i \ne j$, assign an integer randomly selected between 1 and 10 to A[i,j]. Ensure that A[i,j] = A[j,i] since you need to create an undirected graph. Display the generated adjacency matrix.

   For the created undirected complete graph, write a program to find all pairs shortest paths using Floyd's algorithm. Print all pairs shortest paths and their lengths. Finally, save your source code in a file and name the file as *yourlastname_yourfirstname_pa3_floyd.cpp*.

   - Note 1: You are supposed to **implement the algorithms correctly for random undirected graphs** as described above. If your program produces correct results for some graphs but doesn't for other graphs, you will get zero.
   - Note 2: For grading purposes, don't pass any parameter to your main() function. You will lose 10% if you violate this requirement. If everybody follows this rule, grading may finish earlier.

3. [10%] 10% of the grade will be based on *good coding style* and *meaningful* comments.

All programming must be done using **C or C++ in Linux** where your code will be tested. Create a tar file that includes (1) source code files, (2) a Makefile to produce an executable, and (3) a readme file that clearly describes how to run your code. Submit only the tar file

through the Blackboard. The name of the tar file should be yourlastname_yourfirstname _proj3.tar (Do not use special characters like #, @, or &, because they have caused Blackboard problems in the past.) Suppose that your assignment files are under the directory of /your_userid/yourlastname_yourfirstname_proj3/ and you are under that directory right now. To create a tar file under /your_userid directory, do the following in Linux command line:

```
>cd ..
>tar cvf  yourlastname_yourfirstname_proj3.tar  yourlastname_yourfirstname_proj3
```

To view the content of the created tar file, do the following in Linux command line:

```
>tar tvf  yourlastname_yourfirstname_proj3.tar
```

Finally, read the following policies carefully:

- *All work must represent each individual student's own effort. If you show your code or any other part of your work to somebody else or copy or adapt somebody else's work found online or offline, you will get zero and be penalized per the Watson School Academic Honesty Code (http://www.binghamton.edu/watson/about/honesty-policy.pdf).*
- *To detect software plagiarism, everybody's code will be cross-checked using an automated tool.*
- *Your code will be compiled and executed. If your code does not compile or produce any runtime errors such as segmentation faults or bus errors, you will get zero.*
- *The instructor and TA will not read or debug your code. The instructor and TAs will not take a look at an emailed code. If you need general directions, show your code to a TA during her office hours. The TA will not do programming or debugging for you though. The TA will only help you understand algorithms to be implemented and answer basic questions related to implementation.*