CMP SC 3050 Homework 4
Due: 11:59:59 pm, 11/11/2016

The third homework for CMP SC 3050 is a ***programming assignment***. There are 40 points possible for this assignment and will contribute towards 4% of your total grade. The first part of the homework specifies the exact problems that your submission should solve and the second part describes the constraints that you **must** follow (no exceptions). Absolutely no late work will be accepted.

Specification:

For this programming assignment, one compressed file containing the directory of all of the assignment files should be submitted to Blackboard prior to the due date. Your assignment can be built with a standard Makefile, or as a Netbeans Project. Your program should compile on a UNIX machine.

The assignment should:

- Read in two filenames as input in the command line. The first file is the input file and is going to represent an input to the program. The second file is the output file to which your program will write the expected output. Do not hard-code any filenames in your program – doing so will result in a zero – no exceptions.

- The input file is going to store a **weighted undirected graph** as follows. The first line of the file will represent the total number of vertices, and the remaining lines will be vertex pairs representing edges. We will assume that vertices are numbered as 1,2, …. An edge between two vertices numbered as x and y with weight z will be represented as (x,y,z) (with no whitespaces). In the input file, edges shall be separated by newlines. Two sample input files can be found on Blackboard.

- The output file will contain the output. If the graph in the input file is not connected then your program should output 0 followed by a new line character. If the graph in the input file is connected then your program should output a minimum spanning tree of the graph. More specifically, if the graph in the input file is connected then each line will contain an edge followed by a new line. An edge between two vertices numbered x and y should represented as (x,y). Two sample output files can be found on Blackboard.

Constraints:

- The assignment must be completed in C or C++. Your assignment can be built with a standard Makefile, or as a Netbeans Project. Your program should compile on a UNIX machine.

- The executable built by your Makefile or Netbeans project should be named *pawprint*HW4 where pawprint is your pawprint. Please do not include an executable in the submission, just the project or the Makefile.

- Built-in data structures and external libraries must not be used for this assignment. If the program requires a stack, linked list, or any other structure, it is up to you to provide it. If you have any doubt on whether or not any technique you wish to use is acceptable, do not hesitate to ask. You are, however, allowed to use any library in the C standard. A complete list of the allowed libraries is available at: https://en.wikipedia.org/wiki/C_standard_library

- A moderate amount of error checking and resource management is required. Even if you do some error recovery, you must report errors in the input file. Your program should ensure that each line from the input file is properly formatted, that the file is successfully opened, the file is successfully closed upon reading of the file and that all allocated space is de-allocated at the exit.

- For reporting input format error, please use the enum type provided in input_error.h. Please do not alter the files input_error.h. You need to include input_error.h in your main program. The file input_error.h contains documentation of how to use input_error.h.

- A moderate amount of formatting and documentation is required. Comments should be descriptive and used to illustrate the purpose and inner workings of an algorithm or function; they should not be used to annotate each line or self-evident logic.

- The assignment must be submitted via Blackboard. Please compress the folder before submitting using zip or tar. Please do not submit a rar file.

- The input size is unknown, and the input range is 0 to UINT_MAX. This means that your program should not make any assumptions about the range or size of the input. Assuming an input size will result in a grade of zero – no exceptions.

Timing Constraints

A good algorithm for this homework should run in O(m log n) where n is the total number of vertices and m the total number of edges in the input graph. While we shall not be grading the exact time complexity of your solution, we shall be timing your solution and grading efficiency. As a guideline, on input files each consisting of a hundred thousand edges, your program should run in under a minute of system time to receive full credit for efficiency.

Grading:

There are 40 points possible for this assignment. The grade breakdown is as follows:

- 5 points for error checking and resource management.
- 5 points for general programming style and adherence to the constraints.
- 25 points for correct outputs.
- 5 points for efficiency.
- **Failure to adhere to the EXACT output format will result in a grade of zero – no exceptions**

If the program fails to compile or crashes due to a runtime exception, a grade of zero will be assigned.