**Texel student exercise #1 – Amit Berner**

1. For the problem described in the document my solution is to create 3 types of container structures.   
   The first is a Node structure which contains a key value of the node for example "A".   
   The second is Edge which is a connection structure between two nodes.  
   The third would be a Graph which is a container for both of the previous structures.   
   In order to reach and print the adjacent nodes (the immediate relations of smaller and larger than) each node while initializing would create an iterable object that would contain the neighbors, one for each type of relation and when adding new relation, the object would update accordingly. That way we could access the immediate relations in a O(1) time complexity
2. There are two types of inputs that would not make sense that I could think of, the first kind is an immediate relation violation which means two conflicting rows in the text file such as "A">"B" and "A"<"B". the second type of input is deduced conflict such as "A">"B", "B">"C" and "C">"A". the way that I would handle such input errors is to check the immediate relation objects to check for conflicting input for the first type and for the second use a DFS algorithm to check for a directed graph loops, if a loop is found that means there is a conflict in the input.
3. See attached python file, I used the existing NetworkX python library for the implementation since it already contains the data structures I needed to complete the task. Overall the time order of complexity is determined by the last section of the script where in order to look for all the paths in the graph, a single search in the DFS algorithm called all\_simple\_path is O(V+E) and in the worst case when all the rules are for different nodes (A>B,C>D,F>G etc..) the whole loop section is O(V²(V+E)).
4. Attached text files as tests for the python program and the expected result in the next page.

Positive 1 – output

dd<B<A

F<B<A

C<A

C<L

Positive 2 – output

F<dd<C<B<A

L<dd<C<B<A

Negative 1 – output

Invalid input – conflict type 1 (cannot contain A>B and B>A for given A,B inputs)

Negative 2 – output

Invalid input – conflict type 2 (deduced relation cannot exist)

Negative 3 – output

Invalid file format

E- edge number

V-node number

First loop over the file – O(E)

Second loop – O(E)

Find cycles for DFS algo – O(V+E)

And worst case for the