In this final assignment you are going to submit both a program and a written component; both should be submitted using SVN. The program is going to be your solution to assignment 2, this time re-written (still in Python 2) to satisfy the principles of modular design. The written component is going to be the accompanying documentation—please submit in PDF. You should be working with the same group as you did for assignment 2.

In essence, this time we are going to grade your programming "style." We are going to be looking for the following three standard principles of modular design:

- Separation of concerns
- Abstraction
- Information hiding/encapsulation

In your documentation you should explain what are the modules in your program, and justify your design decision. What were the software requirements and how they were met by your design. Discuss implementation issues, as well as the testing you did with sample "runs," i.e., the contents of the link-state database as they change according to the `add`, `del` and `tree` commands. Finally, include a short (say, 2 pages) manual that would accompany your software.

More precisely, follow the guideline below to complete your documentation:

Part I Software Requirements Specification

1. Introduction
   - Purpose of software system
   - Background (mention routing, OSPF, etc.)
2. General System Description
   - Major system capabilities
   - Major system constraints (e.g., the link-state database is not "aware" of the network topology outside the autonomous system)
   - User characteristics (e.g., intended users: it is supposed to be an interface for other routers to communicate through—mention "robustness", and "reliability" when faced with "garbled" commands)
3. A list of numbered (labeled) requirements (here list all the requirements and give them numbers/labels so that you can refer to them when discussing later your modular design)

Part II Software Design Documentation

1. Module Guide
   - System decomposition diagram (for this part, see the Parnas article on "Rational Design"; pg. 253, section B.)
   - Description of each module using a consistent template
   - Traceability for requirements to modules (this is where you will be using the labels for your requirements)
   - **Tips:** when decomposing a system into modules, ask yourself these questions:
     - is the decomposition made by "secrets," not by a sequence of steps? (answer should be 'yes')
     - are the secrets nouns, not verbs? (answer should be 'yes'—in our case, however, this does not apply very strongly. The reason is that it is a simple programming project, where the most natural decomposition into modules is according to "control flow")
     - does the same secret appear in multiple modules? (answer should be 'no')
     - are the modules not too small (larger than a single function), yet not too big? (answer should be 'yes')
2. Justification for design
   - is the design feasible?
   - does the design implement the requirements?
   - traceability for requirements to modules

Part III Implementation Issues and Testing

Part IV User Manual