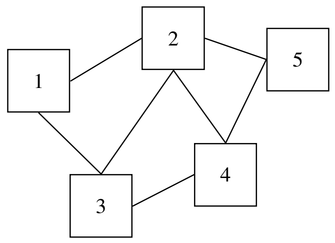
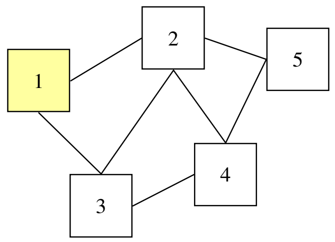
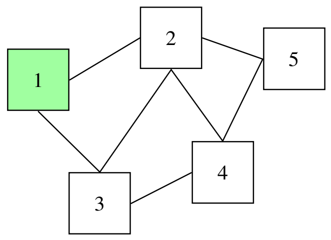
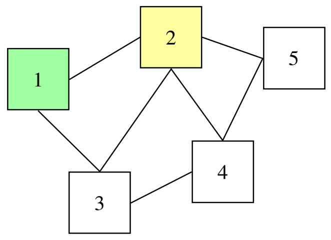
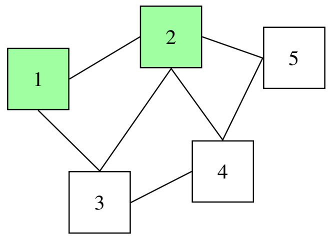
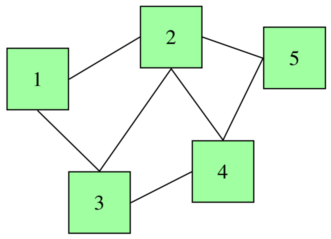
TOP-DOWN AND BOTTOM-UP PROGRAMMING

Top-down and bottom-up programming refer to two different strategies for developing a computer program. ***Top-down programming*** starts by implementing the most general modules and works toward implementing those that provide specific functionality. ***Bottom-up programming*** implements the modules that provide specific functionality first and then integrates them by implementing the more general modules. Most programs are developed using a combination of these strategies.

Both strategies are based on ***incremental development*** – the process of building your program piece by piece. Both strategies employ ***unit testing*** – testing each individual piece before moving on to the next.

Example Here's an illustration of incremental development with unit testing.	
This picture represents the design of a program with 5 major components, all of which have been designed but none implemented.	
With incremental development, you first implement component #1.	
Next, you fully test it.	

Continue by implementing component #2.	
And test it. Continue until all components are implemented and tested.	
Continue until all components are implemented and tested.	

Incremental development with unit testing is an attempt to artificially impose proximity of cause and effect onto the software testing process. Hardware failure obeys ***the law of proximity of cause and effect***; when hardware fails, the cause of failure is in close proximity to the symptoms of failure. For example, if an airplane's wing is vibrating excessively then it is probably caused by some structural problem in the wing. Software failure, however doesn't obey this law because there can be tens of thousands of lines of code between an error and its true cause.

Example	
Consider this state of development where component #1 has been implemented and tested and component #2 has only been implemented. Any errors that arise during testing are likely to be within the newly implemented component.