Develop Integration Test Plan Elements

Goals and motivation for the assignment: The purpose of this assignment is to help the student gain insight into, and experience with, the kind of analysis that can be done prior to coding, and the nature of the planning that can be done for integration testing.

What is the assignment?
In previous assignments you worked on providing details about a complex software system of interest to you. In the previous assignment you defined the interfaces to a number of modules that were to be of pivotal importance in supporting the implementation of your system. In this assignment you are to think about how to carefully and critically evaluate how compatible your modules are with each other by determining whether your modules integrate with each other correctly and effectively.

SPECIFICALLY:
1. Create a Test Harness for two of your modules.
   a. Develop a piece of pseudocode that uses two of your proposed modules to satisfy a system requirement. Your pseudocode must be written in such a way that the outputs of one module are used as inputs to the other. This piece of pseudocode will thus function as the specification of a Test Harness, namely a driver that will then provide the context in which some aspect of the compatibility of the two different modules can be evaluated.
   b. Derive the Control Flow Graph (CFG) that represents the Test Harness pseudocode.
   c. Create an assertion and place it in the pseudocode. Indicate which of the originally stated (in Assignment #2) requirement(s) the assertion is intended to help verify.
   d. Specify at least two sets of test cases that are to be run through the Test Harness for the purpose of verifying its assertion.
   e. Devise a Finite State Automaton (FSA) that captures one of the requirements originally stated (in Assignment #2) for your system (Suggestion: perhaps a robustness or safety requirement would be most appropriate here). Your FSA definition should include its event alphabet. This event alphabet must be drawn from the set of events each of which corresponds to the execution of some method associated with one of the modules used in your Test Harness. Be sure that your FSA indicates which state is the start state, which state(s) is/are accepting state(s), and should also include a violation state that corresponds to the possibility of a sequence of events that violates your requirement.
2. (For 620 students only): Be sure that your Test Harnesses includes at least one loop. Place your assertion for that Test Harness inside of one of the loops to serve as a Loop Invariant Assertion. Then perform a symbolic execution of a path that iterates the loop and use that symbolic execution to demonstrate that your assertion is indeed an invariant.