Comp 170 Project Proposal.

Changes and options to the textbook, section 18.9.

1. Teamwork: optional, you may be alone or teams of up to 4. Teams must do more work than individuals. See the text for roles on a team. Everyone will be a developer.
2. Project Name: you need a name.
3. Weekly Report: all need to do and turn in (one per team)
4. What Will You Do?
    a. You can do the project in the text
    b. You can do anything else
    c. You can do extensions to something others have begun (must be C# and console I/O)
5. Technical Requirements. Your finished project must include *at least* these items
    a. At least 3 classes, two that work together plus a separate class containing Main (like BookList assignment)
        i. Use of appropriate kinds of instance variables, initialized in constructors.
        ii. Implementation of constructor(s) and standard getter/setter methods (if needed) plus other methods.
        iii. Creation and use of objects from these classes in the Main method.
    b. At least one of your classes must use a container (array, list, dictionary) to include other objects.
    c. Well structured code in all parts (classes, methods, Main). Use function/subroutines/methods to divide up the code into understandable chunks. Use appropriate design such as a Top - Down or stepwise refinement approach Avoid long, unstructured code in any single method.
        i. Use of if/else and while/for/foreach statements plus functions/methods, possibly nested.
        ii. Functions/methods must have appropriate parameter and return types.
        iii. Use of **fio** and **ui** libraries as appropriate to interface with the user at the Console and with Files.
    d. Use of Files to hold text or numeric data rather than putting lots of strings or numbers in program source code.
    e. A user interface using the Main method that allows full demonstration and testing of the app.
        i. special user input that causes Main to run the tests or turn debug on.
        ii. Tests created in Main should exercise (all) code paths in all classes, including error cases ("safety").
        iii. Optional: create static test functions in each of the non-test classes that can be called from Main to test those by themselves (unit tests).

f. All code to be well written with readability in mind. Use the style conventions introduced in class and in the text. The goal is code that others can read and understand (and possibly modify without your help).

Project Proposal Outline

Complete these items for your project plan:

Project Title (make it interesting)

My App Definition

A paragraph or two describing what the app does and why it's interesting. Some also call this your "elevator pitch" - if you happen to bump into someone in the elevator and you have 30 seconds before they get out, how do you sell them on your great idea for an app.

Team or Individual
If you plan a team, list all members and possible roles.

Design

What are the key C# components you plan to use? What kinds of input and output will you need from the user or from files/

What will the user interface look like? You can mock it up in text to give an example use case.

Initial ideas for classes. These are often the nouns in the App Definition. A couple of ideas of things the classes should be able to do (these may be the verbs in the App Definition).

Anything you are not yet sure how to do?

Schedule

Make a 4 week plan to do the work. List key activities for each week:

Week 1 (Monday start date) Key things to do

Week 2 (Monday start date) Key things to do

Week 3 (Monday start date) Key things to do

Week 4 (Monday start date) Key things to do

Testing

How will you test your app? Do you plan to have anyone other than you try it (you should!)? If you want to try out some software engineering concepts, keep a list of each of the bugs your find while testing and how long you spent testing. Make that part of your project demonstration.

Version 2.0 6Nov14 (expanding technical requirements, incorporating work of Dr. R. Yacobellis)
Version 1.0 (original) Fall 2014