# :

What you handed in appears to be identical to the starting project. If I missed the work that you did please let me know ASAP (email is great). If you handed in the wrong thing you can still hand in your work by using an extension. To do this you must upload your work to StudentTracker into the “Assignment 3 Revision” slot within 24 hours of when I emailed out this feedback. If you do this please email me to let me know so I can go find and grade it.

And, if you don’t want to do either of the above two options I’m going to leave this rubric here to help guide your work on the final, revised version.

*Overall:*

* Looks good.

*Working Around Problems:*

* This looks good - you have both a plan to work around the problem ‘right now’ and you’ve got some ideas what you might do in a work environment.
* I did not find a comment, located at the top of the Program.cs file, that explains what the compile-time problem was and how you worked around it AND how you'd react to this situation in a real-life/professional work environment. (-6)  
  *HINT:* If you need to do this for the revision you should probably re-download a brand new copy of the starter project and see why that new copy does not compile.

*Book-Tracking Application:*

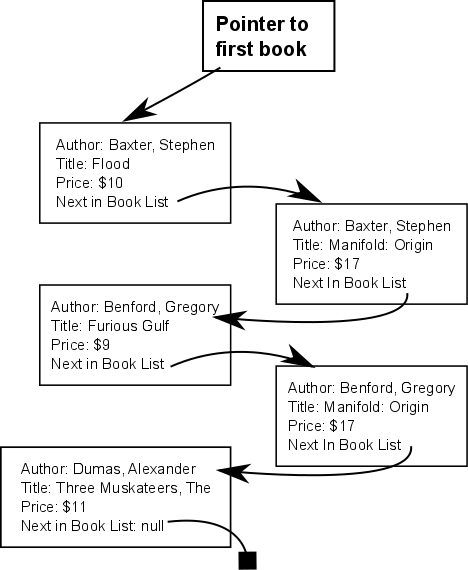
* You haven’t filled in error-handling code (-15)  
  In the Add and Remove options, make sure to give the user confirmation feedback when things go ok, too.
* In the Add and Remove options, make sure your error-handling code gives the user confirmation feedback when things go ok, too (-5)
* Looks good.

*Class for a node in the ‘Book List’:*

* This is missing entirely. (-30)  
  (You should model the individual books using a class)
* Since the class is a private nested class you can (and should!) make the instance variables public. (And remove any get/set methods, too) (-3)
* Put a constructor on this, so you can create Book objects in a single line. (-3)
* You don't have a string field that stores the author. (-3)
* You don't have a string field that stores the title. (-3)
* You don't have a field that stores the price. (-3)
* Remove the ‘book’ prefix from the instance variables. You know that it’s part of the book object because they’re instance variables on the Book class. (-3)
* Add a Print method so you can call it from the MultiLinkedList.Print method. (-3)
* The Book's Print method should live here, not on the MultiLinkedList class. (-3)
* Looks good.

*Multi-List class*

* The basic structure for the lists in this assignment is that the MultiLinkedList has a reference to the first book in the collection (in the picture below, the author is Stephen Baxter). That object also contains the title of that book (“Flood”) and the price ($10). The last thing it contains is a reference to the next book in the list (in the picture below, this is Stephen Baxter’s “Manifold:Origin”, $17). (-30)  
  Your approach appears to use a different approach, instead.



* You should have a single ‘firstBook’ instance variable (you do NOT need the firstTitle variable) (-10)
* Add: don’t create the new Book object until you know that you can use it (-3)
* Don’t create variables that are aliases for the first book reference. (-3)  
  (Just use firstBook directly, so it’s clear that you’re intentionally doing something with the first book in the list)
* Add: Move the code for adding the new node to the very front of the list outside of the loop so that it only runs once (instead of checking for this every time the loop runs) (-6)
* Add: You don’t have any code to add the book to the very front of a non-empty list.  
   (-6)
* Add: You don’t have any code to add the author to the middle of the list (-6)
* Add: You don’t need a separate case for a single-item list. Merge this into your ‘middle-or-end-of-the-list’ logic; doing so will simplify your code substantially.  
   (-6)
* Add: refactor your ‘add book to the book list’ code so that you don’t have to copy-and-paste this (i.e., put it into a method, or else rearrange your code so that you only need to have a single copy of the code) (-6)
* Add: You don’t detect duplicates (-6)
* In Add, incorporate the check for a duplicate into the ‘adding a book’ loop. This way, you won’t have to go through the list twice. (-3)
* Add: This will crash if you add a book to the very end of the list (-6)
* You don't provide a Print method that prints out books in the list pictured above. (-15)
* You don't provide a RemoveBook method. (-30)
* Remove: This should return an ErrorCode and NOT print anything (-3)
* Find a way to implement the Remove method that doesn’t involve creating a new book object each time you want to remove something – Either another version of the Compare methods on the book class, or else an instance variable on the List class that you re-set each time. (-3)
* In Remove, refactor the “is the book identical to author & title” to a method on the book class (or re-use the Book.CompareBook method, instead of duplicating this again here. (-3)
* Remove (and elsewhere): on lines you create a local variable named **copy**. This is the same as firstBook on these lines. It’s confusing to create a new name for something you’ve already got, so on these lines just refer to firstBook directly (instead of using the local variable you just created) (-6)
* In the Remove method, you should look for an exact match in the list (i.e., same title AND same author), then remove the exact match from the list. (-6)
* In your Remove method, you can (and should) end before traversing the entire list, since the lists are stored in sorted order. (-6)
* Remove unused/ commented out code before handing this in (-3)
* Except for methods whose purpose is specifically to interact with the user (such as PrintByAuthor/Title), this class shouldn't produce any I/O. Instead, return a value to the calling method, and then use that to figure out what main should print.) (-3)
* Looks good.

**Grade (out of 130):**

**To calculate your grade:** add up all the (-1)'s and (-6)'s and (-X)'s, to get some negative number, then take that from the total to get your grade. For example: If the total points available for the assignment was 100, and you had the following penalties: -3 + -6 + -6 🡺 -15, so the grade would be 100 -15 = 85.

Why do you have to do this? Because this is only version 1, and so you won't really get your 'real grade' until you hand in the revision. Sometimes the grade on this first version appears really low (especially if you left out a whole section), and so I want to give people feedback, but try to avoid spooking people. Keep in mind that if you don't hand in a revision, this will be your final grade.

Why do you have to do this? Because this is only version 1, and so you won't really get your 'real grade' until you hand in the revision. Sometimes the grade on this first version appears really low (especially if you left out a whole section), and so I want to give people feedback, but try to avoid spooking people. Keep in mind that if you don't hand in a revision, this will be your final grade.

Numbers that are 'greyed out' don't count (i.e., the item is there for informational purposes, to preemptively give you feedback for your revision), but these don't actually represent points that you've lost. Example of a 'greyed out' item:

You didn't do X. (-3)

**Note**: Please note that if any of the above errors are duplicated within your code, you need to fix ALL INSTANCES of the error, even if it's not specifically listed above, in order to get the points.