Rules:

1) The test is this WORD document with detailed instructions. It is very much like a homework, just **finish** a Visual Studio solution per this document. **I have provided a starter project with all the classes already created.**

2) The correct output could be worth zero points, yet the incorrect output might be worth 25 points. How can that be? If you get the correct output but do not do it the way I prescribe, you will get no points. If you attempt the project the way I prescribe and one small mistake prevents it from working, you might get 25 or more points. So please read this WORD doc carefully.

3) When you are done, y**ou cannot zip a file while Visual Studio still has it open**. Close Visual Studio, then ZIP.

4) Also required: After you submit your zipped project to Canvas, open your project again and cut and paste all the code in the Program.cs file into Canvas email and send it to me. In a class of 25 students, it seems at least one student submits the wrong project or leaves out some files every final. If you have mailed me the code, it won’t be a problem. If you haven’t emailed me the code, and you zipped the wrong project, then you failed the final.

=====================================================================

**The big picture:**

The test has 3 phases, each worth 10 points.

* Can you make a parent and child set of classes work correctly?
* Can you make sort work?
* Can you make a delegate method work? (Just a delegate method, NOT an Event)

Here is what you need to do. .

1) Edit the Movie class. Give it 2 properties: String **Title**  int **Year**

add an overridable method, GetDescriptionString

this method does not take in any parameters, but should return the string “There is a bug in this program”.

2) Edit the 2 classes, Comedy and Horror. For both of them:

Make them children of the Movie class.

Write a **constructor** to set the 2 properties that they inherit from their parent.

Override the parent’s GetDescriptionString method so that the child method returns a string

Title: value-of-Title-prop Year: value-of-Year-prop

*So it might return a string like:* Title: Moonstruck Year: 1968

Assuming that movie object’s properties were set to “Moonstuck” and 1968

**Do not** put a Console.WriteLine in this method! This method returns a string, it does not write out a string.

3) Your Main method already creates 6 movies, create a List of type Movie named myList.

Add the 6 movies to that list

4) Next write out to the console information from each of those 6 movie objects, using a **foreach loop** which writes data returned from calls to the GetDescriptionString method that these child objects have. (If you cannot get a foreach to work, use a for loop instead, you will lose 1 point, but at least you will be able to complete everything else.) *(The program should run at this point, +10 of 30 points)*

5) When that works, extend the Movie class to support the IComparable Interface **for objects of type Movie**. Do the comparison using the Year property.

6) Add 2 empty Console.WriteLine(); statements for spacing and then use the List Sort method to sort the list. Then cut and paste the code from step [4] to write the list out again, but this time it should be sorted by Year. (see my sample output below)

*(The program should again run at this point, +10 more of the 30 points)*

8) Add 2 empty Console.WriteLine(); for spacing.

Look at the Magic class and the delegate definition that precedes it to understand what it does. You **should not make any changes** to the Magic class.

Back in Main, instantiate an object of type Magic.

Now add another foreach loop at the end of Main. In the foreach, call the Transform method in you Magic object, passing in, for each movie, the string returned for calling each object’s GetDescriptionString(), as well as a new method you have to write named AllCaps.

9) Now write the AllCaps method. Its signature must match our OurDelegateDefintion. All the method has to do is take in a string and return it as all capital letters. Just 1 line of code should do it. (see my output).

Below is what your program output should look like when you are done.

