CIS 330: Project #3E
Assigned: May 16th, 2018
Due May 23rd, 2018
(which means submitted by 6am on May 24th, 2018)
Worth 3% of your grade

Please read this entire prompt!

Assignment:

Make your data flow network design do demand-driven execution.

Turnin: a tarball of your source code, plus a Makefile

Please note: a new main3E.C and a new Makefile are on the class page.

== Background ==

In 3C , the main function in main3C.C called Execute for every filter in the pipeline, as well as the reader. Data flow networks typically use demand-driven execution. This means that a program requests that one object (i.e., Image) be up-to-date. The object is typically at the bottom of the pipeline. For that object to be up-to-date, it needs for the Source that generates it to Execute. If the Source that generates it is a filter, then the filter can't execute until its inputs are up-to-date. So the filter would prompt its inputs to get up-to-date. In this way, the request to get up-to-date propagates all the way up the pipeline. When it gets to a file reader, the data is read. Then the filters can execute, confident that their inputs are up-to-date.

Saying it another way, you can imagine an "update" request flowing from the bottom of the pipeline. As the input to each filter establishes it is up-to-date, the filter can execute. This effectively makes data "flow" down the pipeline. And it does it all from a single Update call (which in turn spawns many more Update calls and many Execute calls).

== Changes to your program for data flow ==

(1) make the Execute method in Source be protected
(2) Add an "Update" method to the Image class.
    a. Hint: you will need a virtual method named Update for Source as well.

There are other changes that you will need to make that are not described in this document. But do not modify main3E.C in your final submission… you need to modify your data flow infrastructure to execute successfully from the single Update.

== Other change ==

Make the Image data member of your Source (the “output”) be protected if it is not already. Also make the Image * data members of your Sink class (the “inputs”) be protected if they are not already. You may still allow access to these data members via methods (i.e., “Image *GetOutput(void) { return &output; };” as a public method)

Also, make sure all files follow the traditional file structure:

- Headers contain class definition only
- Methods in .C
- Exceptions for “inlines” OK if the function is one line of code (normally getters and setters)