CIS 330: Project #3H
Assigned: June 3, 2018
Due "June 8th"
NOTE: I will not apply a late penalty if this project is submitted after June 8th. However, all work for the course must be submitted by Weds June 13th (6am Thursday June 14th). No work will be accepted after June 13th – 0% credit.

Worth 11% of your grade

Please read this entire prompt!

You will be running the 3T tests you wrote, plus additional tests designed by previous students in this course. Every regression test has its own directory. The directory has source code called "stress_test.C" which contains their pipeline. It also has the correct output. If the program should throw an exception, then there is a file in the test's directory called "exception" with the text "Exception found!". If the program should generate an image, then the file in their directory called "checksum" with the correct checksum.

Directions:

(1) Download Proj3H.tar from the website.
   a. tar xvf Proj3H.tar
   b. cd 3H
   c. copy your source code into the sandbox directory

(2) run the tests and confirm you get the same answer
   a. run_all # runs all tests, declares passes and fails
   b. run_one # runs a single test … used to debug a mismatch
   c. run_one_w_image # runs a single test & creates its output image
   d. Note: not all tests submitted were valid … you should only worry about the tests on "valid_list". "run_all" automatically takes this list into account.

What to submit:
(1) a tarball of your final source code (via Canvas)
   a. cd sandbox
   b. make clean
   c. tar cvf my_proj3H_code.tar *.[Ch]
(2) a screenshot of you calling "run_all" and it stating how many valid tests you have passed.

We will grade this by running on ix-dev. Make sure your code works on ix-dev.

Scoring rubric:
0 incorrect: 11 points
1-5 incorrect: 9 points
6-10 incorrect: 7 points
11-half correct: 6 points
< half correct: 5 points

== What to do if your code doesn't match up? ==

You ran a test and feel like your program is running right? The first step is to run "run_one_w_image", look at the output. Then look at my baseline image at http://ix.cs.uoregon.edu/~hank/330/projects/3H/<testname>.png.
Or
http://ix.cs.uoregon.edu/~hank/330/projects/3H/<testname>.pnm.

Note: test "andy1" is a tough one. Debugging that will require some thinking and some new code. I expect that most solutions to that program will be hack-y. Not "if andy1 then throw exception hacky", but probably involve some global variables or something. Also note that if you pass that test, I intend to look at your code and see what you did. If it is super hacky, I'll likely mark the test as incorrect.

Note 2: Shrinker is a bit ambiguous when it comes to odd dimensions. If an image is 3x3, then it would be legitimate to create a 2x2 output. My previous prompts are ambiguous on this front. That said, almost everyone creates a 1x1 (since 3/2 is 1 with integer arithmetic). So let's make that our convention and make sure your Shrinker works that way.

Note 3: as you modify your code, you may create problems elsewhere. Our system can get tricked, as we add a file called "pass" to a directory each time you pass its test. So you might get a "pass", but then modify your code and it would stop passing. So: I recommend you do "rm */pass ; ./run_all" before submitting. That will double check that none of the tests stopped working after you made them work the first time.