Quiz 4 (Open book, 45 minutes, 140 total points)

Please build your code based on the sample code in the svn repository of
file:///home/jxue/csci251Repos/imageProcess on the server turing.cs.olemiss.edu. You
might need to use the following Linux commands in putty:

- ls       - display files and sub-directories in the current path
- pwd      - display the full name of the current path
- cd ..    - go to the path that holds the current path
- cd csci251Repos - go to the csci251 repository
- cd imageProcess - go to the imageProcess folder inside the repository
- svn update - synchronize with the svn repository in the current path

Please use windows software Gimp to browse pgm images.

Part I. Required problem (100 points)

Please implement a Fortran 95/2003 program that generates an output image that is the
horizontally mirrored input image as shown in Figure 1. Please implement the image
processing algorithm in a subroutine named mirror, it should have 4 arguments including
an input array, an output array, and their column and row sizes.

![Image of left and right hands](image.png)

Figure 1 Example input and output images of the image processing tool.

Please include both the main program and the subroutine in the same Fortran source code
file named processPgmMirror.f95. Please include your name, student ID, and section
number in comments in the source code. Please send processPgmMirror.f95, along with
your output image using the following command inside Putty:

```
mutt jxue@cs.olemiss.edu -s "Quiz4" -c yourEmailAddress -a yourOutputImageFile < processPgmMirror.f95
```

Optional problem is continued on the next page...
Part II. Optional problem (40 points)

In csci251Repos/imageProcess repository, you can create a pgm image down sampling tool using the following compiling command:

```
f95 processPgmDownSample.f95 downSample.f95 -o downSample
```

The following example shows the results of calling ./downSample iteratively:

![Figure 2.1 Output images generated by calling the following commands:](image)

(a) ./downSample < johnyDeppsHand.pgm > jdhsmaller1.pgm
(b) ./downSample < jdhsmaller1.pgm > jdhsmaller2.pgm
(c) ./downSample < jdhsmaller2.pgm > jdhsmaller3.pgm

The smallest image in Figure 2.1(c) is lack of details. If we need to create a thumbnail version of an image, this method yields poor representation of the original image. A robust down sample tool yields the results as shown in Figure 2.2.

![Figure 2.2 Output images generated from robust down sampling in three iterations.](image)

The robust down sample tool can be built using the combination of imageFilter and downSample subroutines in two steps. You can use the following kernel matrix

\[
\text{kernel} = \begin{bmatrix}
0.05 & 0.12 & 0.05 \\
0.12 & 0.32 & 0.12 \\
0.05 & 0.12 & 0.05 \\
\end{bmatrix}
\]

to filter the input image first, then down sample the filtered image. Please write a program named processPgmPyrDown.f95 to implement the algorithm. Your source code needs to include your name, student ID, and section number in comments. Please submit your source code and the output image from the 3rd call of your program as follows:

```
mutt jxue@cs.olemiss.edu -s "Quiz4Opt" -c yourEmailAddress -a your3rdIterOutputImageFile < processPgmPyrDown.f95
```