Suppose we have a digital image that was taken by a satellite and that the information from the satellite is in a two-dimensional array of integers with each element having a value from 0 through 9. Each number in the array is a *pixel* (picture element). Due to transmission errors, random noise has entered into the image. Another problem with the image is that detail is blurred by the many numbers used in the representation.

We would like to enhance the image in two ways:

1. reduce the amount of noise
2. increase the contrast by reducing the number of symbols in the image

**Reducing Noise**

Noise can be detected when the value of a pixel is very different from its neighbors. For example, in the image section

24456

34187

48899

the pixel with the value of 1 can be assumed to be noise since its value is so different from its neighbors.

If a pixel has a value that is different from its eight neighbors by 3 or more, it can be assumed to be noise. We will ignore the pixels on the edge of the picture.

A noisy value should be replaced with the rounded average of its eight neighbors. In the above image section, the 1 should be replaced with a 6:

*value* = (4 + 4 + 5 + 4 + 8 + 8 + 8 + 9) / 8 = 6

**Increasing Contrast**

Humans discern shapes better with fewer symbols. By reducing the number of symbols, we can increase contrast in the image. To reduce the number of symbols, we will lump the numbers 0 to 3 into one category; the numbers 4, 5, and 6 into another category, and the numbers 7, 8, and 9 into a third category. Set the numbers 0, 1, 2, and 3 to 0; set 4, 5, and 6 to 1, and set 7, 8, and 9 to 4.

Image data are stored in *image.dat*. The file consists of data for a 20 x 77 array.

Name your source file *prog1.c*

You must define the following functions:

***void readImage(int image[ ][COL\_SIZE], int rows, int columns);***

- reads the image data from standard input into the two-dimensional

array that is 20 rows by 77 columns.

***void enhance(int image[ ][COL\_SIZE], int rows, int columns);***

- invokes the *reduceNoise()* and *addContrast()*

functions to enhance the image.

***void printImage(int image[ ][COL\_SIZE], int rows, int columns);***

- prints each element of the array to standard output as a

single-digit integer, with no spaces between elements.

displays the array in two-dimensional form.

- although the original file contains spaces to make it easy

to read the data, this method should not include any spaces

between the digits.

***void reduceNoise(int image[ ][COL\_SIZE], int rows, int columns);***

- checks each element to see whether its value is noisy.

If it is, replace the value with the average value of its

eight neighbors. Do not check elements on the edge. A value is

noisy only if it differs from all eight of its neighbors by 3 or

more.

***void addContrast(int image[ ][COL\_SIZE], int rows, int columns);***

- replaces the values in the image: 0 replaces values 0 - 3.

1 replaces values 4 - 6, and 4 replaces values 7 - 9.

***int main ( )*** - should be the first function in the file.

- declares the image array (20 x 77). use the defined constants.

- invokes readImage

- invokes enhance

- invokes printImage to display the enhanced image.

**Getting the files for this assignment**

1. Use *mkdir* to create a prog1 directory for program 1. You may name your directory whatever you want.
2. **cd** to the prog1 directory.
3. Download the [prog1.tar](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog1/Public/prog1.tar) file to your prog1 directory. You can save the tar file from your browser, or you can directly copy this file over the network using the scp utility:

***scp access.cs.clemson.edu:/home/rlowe/public\_html/cs1070/programs/spr18/prog1/Public/prog1.tar .***

    Note the "**.**" at the end of the scp -- it is important -- meaning extract to the "current directory".

1. Use the command below to extract the files in prog1.tar

***tar -xvf prog1.tar***

After you untar the files, you may delete prog1.tar

Your prog1 directory should contain the files *prog1.h*, *image.dat*, and *makefile*. *prog1.h* contains named constants and the function prototypes for this assignment. You will need to *#include "prog1.h"* in your prog1.c file.

The *make* utility helps you to build and manage projects. We will discuss it later.

**Compiling and Testing Your Program**

1. Use the following command to build the executable, *p1*

make

1. Test the program using

./p1 < image.dat   
where image.dat is the name of the file that contains the image data.

If your code is correct, you should see an interesting image.

**Submitting your files**

Submit ***prog1.c*** electronically using the [Handin](https://handin.cs.clemson.edu/)

It is your responsibility to make sure you have submitted the correct file. Check your Handin folder after submitting the file. There is a 10-point penalty for using the wrong file name and a 10-point penalty for submitting the file to the wrong folder.

**Requirements**

1. For this program, you must work individually. You may discuss the problem with classmates, but at no time should you discuss code in any form. You may not show another student your code, you may not send your file to another student, you may not look at another student's code, you may not tell another student what to type, you may not ask the help desk workers to help you with your code, you may not ask the tutor to help you with your code.

Any evidence of cheating will result in a grade of **-100** for all students involved. If you have questions, you should check with me or a lab instructor.

1. Your program must adhere to the [program standards](https://people.cs.clemson.edu/~rlowe/cs1070/programs/standards.html) and [program requirements](https://people.cs.clemson.edu/~rlowe/cs1070/programs/requirements.shtml). Violations will lead to deductions. In particular,
   * no more than one statement may be written on a single line; this includes declarations.
   * use reasonably descriptive names for variables and functions. no abbreviations allowed, unless universally known.
   * code lines should not extend beyond column 80.
   * indenting should be consistent with logical nesting; indent 3 - 4 spaces.
   * do not use a **break** or a **return** statement to prematurely exit a control structure.
   * a brief description must precede each function.
   * no function should be longer than 30 lines of code, excluding whitespace.
   * a function should have **at most** one return statement.
   * your program should compile without any warnings with gcc -Wall
   * diagnostic / debug prints should be disabled/deleted in the final submission.
2. The program is due by 11:59:59 pm. Programs must be submitted on time. You are granted one "late pass" which allows you to submit one program 24 hours late without penalty. Subsequent late programs will not be graded, and a grade of **zero** will be recorded.
3. Programs must compile. Programs that do not compile will receive a score of **zero**.

**If you use the math library, you must include -lm during compile**