CS 350: COMPUTER SYSTEM CONCEPTS

PROGRAMMING ASSIGNMENT 2: C-PART
ANNOUNCED: MARCH 15, 2010
DUE DATE: MARCH 23, 2010 (IN CLASS, DOCUMENTATION)
MARCH 28, 2010, 11:59 PM (ONLINE SUBMISSION)

MAXIMUM MARK: 100

A copy of this document and other required files for the assignment can be found via the course web page:
http://www.csee.wvu.edu/~adjeroh/classes/cs350/

A copy of the relevant page from the C book is on the course website.

1. MINI DATABASE IN C **(40 MARKS)**
Exercise 11.12, pp. 447: (Deitel and Deitel, *C: How to Program*, 2001)

Please read the extra information on this page carefully, and consider it as you write and submit your programs.

Additional Requirements:

The name of your program file should be **minidb.c**

In addition to the problem description in the book, your program should provide **additional capabilities**:
(1) for searching the database, based on the Tool ID (record number) **and** on the Tool Name,
(2) to save the database to a file, **and**
(3) to load an existing database.

The options should be presented in the form of a menu:
1 Initialize the database
2 Input new records
3 Search for a record
4 Delete a record
5 List all the records
6 Save database
7 Load an existing database
8 Exit

See Problem 2 on Image Enhancement next page …

2. ADAPTIVE IMAGE ENHANCEMENT (50 MARKS)

Note: this particular assignment will form the basis for the OS-assignment, which will involve image manipulation using many processes and/or multiple threads. Thus, an understanding of (and correct solution to) this problem will significantly reduce the time you will need in the OS assignment.

Imagine that you have an image that you would like to enhance in some way, perhaps to expose certain parts of the image that may be too dark, and hence not very visible in the image. Assume you are given an N×N image **I.** You are required to apply some simple transformations (local averaging under specified conditions) on the image to produce an enhanced image **E**. This type of filtering is usually performed during some stages of image and video analysis.

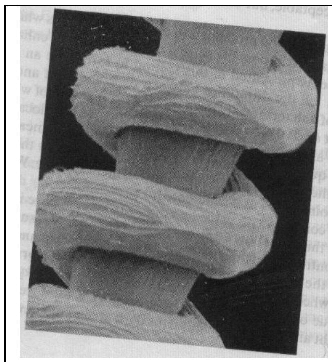
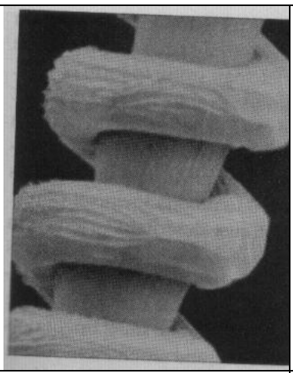
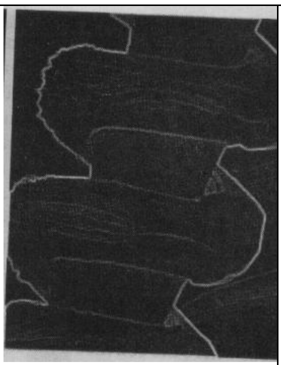
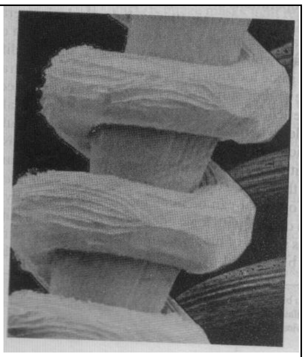
The required transformation is performed using the following steps: Compute the global average M, global standard deviation S, and global median Q of the entire image; Define *n´ n* as the neighborhood block size; assume n is odd (n=3, 5, 7); For each *n´ n* block of the image, we compute the average (m_b) and standard deviation (s_b), of the pixel values within this small block. (This is done on overlapping blocks). Modify the value $I(x,y)$ at the center of the small image block based on the following criteria:

$$E(x,y)=\begin{cases} aI(x,y), \textbf{ if } m_b \le Mc_1 \quad \textbf{AND } Sc_2 \le s_b \ge Sc_3 \\ I(x,y), \quad \textbf{otherwise} \end{cases}$$

For a block of data with n items, $x_0, x_1, ..., x_{n-1}$, the mean and standard deviation are defined as follows:

$$m_b = \tfrac{1}{n}\sum_{i=0}^{i=n-1} x_i \;; \qquad s_b = \sqrt{\tfrac{1}{n-1}\sum_{i=0}^{n-1}\left(x_i - m_b\right)^2}$$

Assume: $a = 2.0; c_1 = 0.4; c_2 = 0.02; c_3 = 0.4$. Sample results from such an enhancement procedure are shown below.

			
Original image I	**Average image**	**Variance image**	**Final enhanced image E**

Note: Images are from R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 2nd edition, 2001. Ignore the slight rotation between the images

Effectively, the image enhancement is performed adaptively, on only selected areas in the image (i.e. only on the areas that deserve to be enhanced, such as areas that are too dark, but still contain some information). Notice how the lower-right part of the image above has been particularly enhanced.

Your program should produce the enhanced version of an original image from the user. For instance, the main function could divide the image into ***n´ n*** smaller image blocks, and then call a function to perform the required operations.
The main function will thus do the following:

i. Read the input image, **I** (**imageIn.ppm**)
ii. Compute the global mean M, global standard deviation S, and global median Q of the entire image
iii. Partition the original image into smaller overlapping subimages (or blocks) each of size ***n´ n***

iv. Send each image block along with *M* and *S* to another function (say an image-analysis function), which then analyzes the image and returns the results. The results will be **four values**: local mean, local standard deviation, local median, and computed enhanced value *E*(*x*,*y*)).

v. Using the returned results, form four corresponding images, (same size as the original), one for each value in the result. Call these the **averageImage**, **varianceImage**, **medianImage**, and **enhancedImage** respectively.

vi. Print summary statistics for the original image, namely, ***M, S,*** and ***Q***

vii. Use the returned results from all the blocks to calculate and printout the following statistics:

1. Overall image mean (i.e. mean of the individual block averages)
2. Overall image median (i.e. median of the individual block medians)
3. Standard deviation of the block averages
4. Average subimage standard deviation (mean of the standard deviation from the individual blocks)
5. Average of the pixel values in the enhanced image

viii. Save the results to respective files: that is, write the average image to a file (**ave.ppm**); the variance image to a file (**var.ppm**); the median image to a file (**med.ppm**); and the enhanced image **E** to a file (**enh.ppm**)

The actual task that the image analysis function will perform is to calculate the average and standard deviation of the input array of pixel values in the image block. That is, the function will:

i. Receive the input image block from the calling function
ii. Compute the mean, and standard deviation, and median
iii. Compute the new enhanced value based on the mean and standard deviation above.
iv. Send back the results to the calling program

Your image enhancement program should be called **imEnhance.c**.

The Input Data

The input data will be an image in **ppm** (portable pixel format) to be provided by the user. (Some sample images are provided in the project directory). Also provided are two functions, one to read images in **ppm** format, and the other to create a **ppm** image from a given data. These functions are in the image library: **iplib2New.c,** available in **imageInfo** directory. A sample program for using the image library is given in the file: **ip03mainSample.c**. You are not required to create your own program for reading or writing image files. But there will be no penalty for doing so :-). However, if you use a different image format, you will have to be able to read from and write to such formats. Also, you do not need to understand how the image read and write functions work. All you need is to understand the interface to the functions – i.e. how to pass parameters to the functions. Assume that images are only gray-level images.

Example Command Line

```
shell$> imEnhance  imageIn.ppm  ave.ppm  var.ppm  med.ppm  enh.ppm  n
```

The parameter ***n*** is the bock size. That is, for a 3x3 block size, we just use *n*=3.

PROGRAMMING STYLE (10MARKS) .

Marks will be given for good programming style, exception handling, error checking, and special considerations for improving the program beyond the stated requirement. Examples here will be handling arbitrary image dimensions, handling color images, etc. The maximum here will be **10 marks**.

Algorithm Design Phase and Documentation. For each problem above, you are required to include a documentation of the algorithm you used to solve the problem, before you started to code. The algorithm will carry about **20% of the mark**, before we start to consider the programs. This is different from the 10% mark for programming style. Submit algorithm design phase as a text file, or a file in .ps, .pdf or Word format. Also, you are required to submit hard copies of your assignment in class as indicated above.

Your programs should be in C, and should be able to run on a Unix /Linux platform.

Please submit your assignment using the usual ***submit*** command:
http://www.csee.wvu.edu/~adjeroh/classes/cs350/assessment/submit.html