TCSS 333 Winter 2016 HW2 Image Manipulation

Due: by midnight Friday Jan. 22 (or with 10% late penalty by midnight Saturday Jan. 23).

In this assignment, you will be working with image files of a particular kind: they will all be bitmap files (with extension .bmp), using 24-bit color and the width and height of the images will always be multiples of 4. Two examples of such files will be posted on Canvas. Your program should also be able to work with other images of the same type. Your goal is to read in such an image file and create two new image files from the first:

- one that is twice as tall and twice as wide (called "big.bmp")
- one that is half as tall and half as wide (called "small.bmp")

Your program should start by reading in a file named "test.bmp". You can find out how .bmp files are formatted by examining them with a hexadecimal editor and comparing what you see to the description at http://en.wikipedia.org/wiki/BMP_file_format. Bitmap files are not all alike and this article provides more information than you need. The most important parts are the two tables labeled "Bitmap file header" and "Windows BITMAPINFOHEADER". The first one explains what is contained in the first 14 bytes of the file and the second one explains the next 40 bytes. If the image is w pixels in width by h pixels in height, then there are a total of w*h pixels. A pixel in a .bmp file is recorded using thee bytes standing for the red, green, and blue color components of the pixel. So an image file with image dimensions 100 by 200 will store 20,000 pixels. That same file will have a total size of:

```
14 bytes    Bitmap file header
40 bytes    Windows BITMAPINFOHEADER
60,000 bytes   width * height * (bytes per pixel) = 100 * 200 * 3
 ----------------
 60,054  bytes total
```

In the headers, you will find

size of the BMP file in bytes
bitmap width in pixels
bitmap height in pixels
the image size

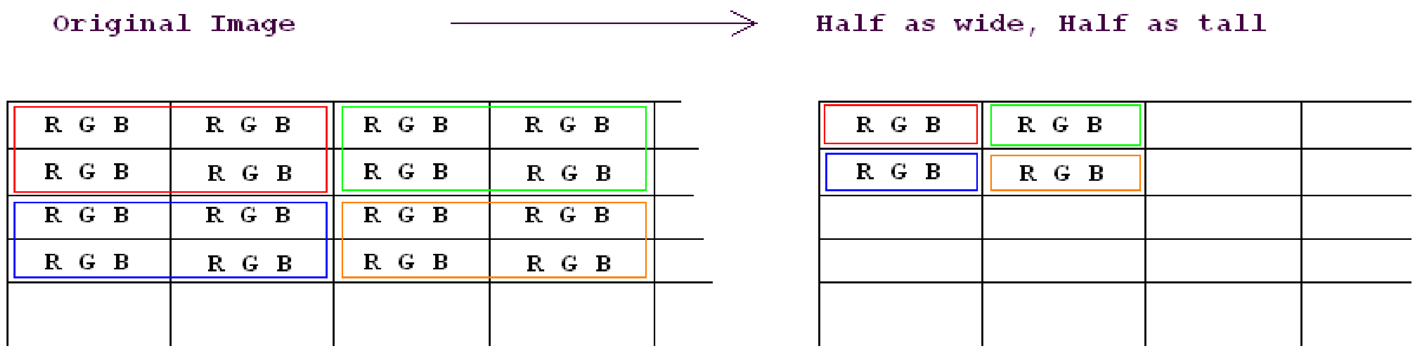
The wikipedia article and a hex editor will help you figure out exactly where they're located. You will need to read those numbers into int variables so you can use them in your program. The rest of the header information should not be changed, so you can simply read those parts into char arrays where those bytes will be stored until you later write them out to your output files.

Once you have finished reading through the headers (using a number of fread's), you will be ready to read the pixel data that makes up most of the file, again using fread. You'll put all that pixel data in one large 2D array.

After reading in the original pixel data, you will need to create two other 2D arrays that will hold your new images (big.bmp and small.bmp). Generally to make a new image, you will have to look at some (or all) of the old 2D array and at the same time place matching data into one of the other 2D arrays.

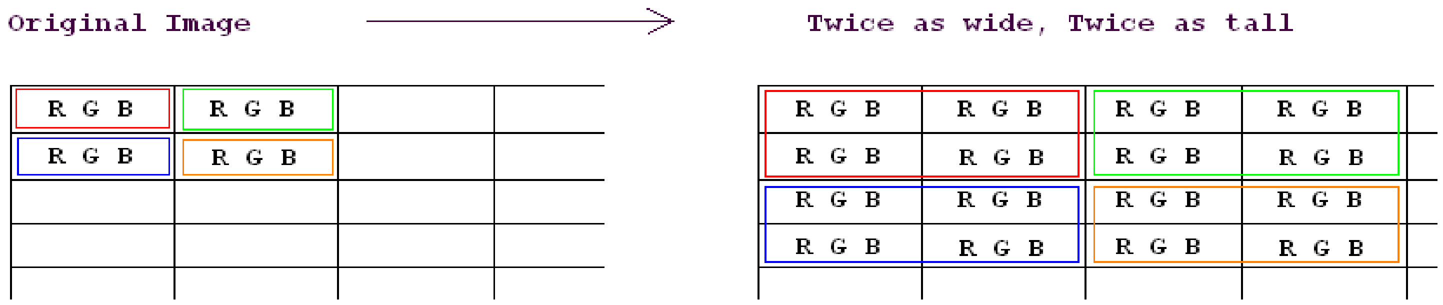
How will you do the shrink operation?

To make an image that is half as wide and half as tall, you will need to leave behind 3 out of every 4 pixels. Imagine that the pixels of the original image are assembled into groups of 4. In the new image, you can only keep one pixel and that one pixel will have a new location, closer to the upper, right-hand corner. Since the original four pixels may all be different, which one will you keep? A good, simple solution is to always keep the one in the upper left corner. (There are 3 other equally good rules.)



How will you do the expand operation?

To make an image that is twice as wide and twice as tall, you will need to create 4 pixels in the new image for every single pixel you find in the old image. The 4 pixels of the new image will be clustered together and each one will have the same color as the single pixel in the old image. This means the new, larger image will not have the same resolution as the original and will not look as "sharp."



After creating pixel data for your new images, you need to write out two .bmp files by reversing the same process you used to read in the starter file, this time using fwrite instead of fread. The new files you make will not be the same size as the original file, so when you write out the headers, be sure to use new values for any numbers that depend on the image size.

As a way of getting started, you should probably try to simply read in all the contents of the starter file and then write out the very same data. This should yield a copy of the original image. If a bmp viewer like Paint is unable to open the file (because it's corrupted) or if the file doesn't look right, then you haven't made a good copy. You should look at the output file with a hexadecimal editor as you do your debugging! If you look closely at integers that appear in the header using the hex editor, you will notice that 4 bytes that make up an integer are saved in reversed order. Instead of 00 00 02 56, for example, the file will contain 56 02 00 00. fread and fwrite will automatically unscramble that for you, so you don't have to. (Be careful if you test with your own images. Even medium sized pictures may kill your program because we are creating large 2D arrays on the stack.) For this homework, you may put all your code in main if you like.