<u>ECE 3331, Dr. Hebert, Fall 2023 Programming Assignment 12, due Saturday 11/18 at 11:59 pm</u>

In this programming project,
(a) you will create a BMP image file and write the appropriate values to the file using write-binary fopen( ) and unformatted fwrite ( ).
(b) you will generate the bitmapfileheader , the bitmapinfoheader in similar fashion to a WAV file header.
(c) You will write the bitmapfileheader to the file using a single fwrite( ).
(d) You will write the bitmapinfoheader to the file using a single fwrite( ).
(e) You will generate pseudorandom values using rand( ) for the RGB values of each pixel in a row of the image and store them in a malloc'd vector. Each pixel in the image will have a unique set of RGB values.
(f) You will write each row of the image to the file using a single fwrite( ).
/**/

Your program will generate a pseudo-random color BMP image of size and filename as specified by the user.
1) First, ask the user to enter the horizontal and vertical size (e.g. M and N).
Instruct the user that the horizontal size must be a multiple of 4 (pixels). Check the entered value, and repeat till the user follows this instruction.
2) Then, prompt the user for the output filename with a bmp extension for the new BMP image (to be stored in the directory from which your program is run).
3) Declare two 1-D unsigned char arrays to hold the BMP file header and the BMP info header.
Use unsigned char bmpfh[14], bmpih[40];
4) Fill-in all bytes of the headers according to the BMP file format handout.
An easy way to fill in the 4 bytes for an unsigned int into an unsigned char array

```
unsigned char bmpih[40;
unsigned int  N=11863;  // for example
bmpih[4] = N%256;  // LSB
bmpih[5] = (N>>8)%256;
bmpih[6] = (N>>16)%256;
bmpih[7] = N/256;  // MSB
```

An unsigned int is made up of 4 bytes B1 (LSB) ,B2,B3,B4 (MSB)
N=B1+B2*256+B3*256*256+B4*256*256*256
N%256 = B1
(N>>8)%256=B2 (shifts B2 to the LSB position)
etc
Use a single fwrite( ) to write out the BMPfileheader and a single fwrite( ) to write out the BMP info header to the BMP file.
5) Use malloc( ) to generate a 1-D vector with the correct amount of space to hold a single row of the BMP image; i.e. rowsize bytes. These bytes are RGB values (in the order BGR) for each pixel in a row.
<u>Remember that the number of bytes in a BMP image row (rowsize) must be an integer multiple of 4-bytes.</u>
For example,
if the image is (width,height)=(3,4), then each row of the image is 3 pixels, whose BGR values occupy (3 pixels)(3 bytes per pixel)=9 bytes.
But since the image data for each image row of a BMP image must occupy an integer number of 4-byte words, you would have to zero-pad 12-9=3 bytes of zeroes to the end of the image data for each and every row.
Therefore, the rowsize=12 and the size of the BMP file would be 14+40+(4)(12)=102 bytes.
The BITMAPFILEHEADER for the 3x4 BMP image would therefore have the value 102 in bytes 3-6 (Size).

bytes 3-6 (4) (**Size**)	total # bytes in BMP image-file=102	102, 0, 0, 0	LeastSigByte 1st; MostSigByte last

The BITMASPINFOHEADER would have Width=3 and Height=4

bytes 5-8 (4) (**Width**)	Image-width in pixels=3	3, 0, 0, 0	LSByte 1st
bytes 9-12 (4) (**Height**)	Image-height in pixels=4	4, 0, 0, 0	LSByte 1st

and ImageDataSize=(4)(12)=48

bytes 21-24 (4) (**ImageDatasize**)	Total # bytes in imagedata =48	48, 0, 0, 0	LSByte 1st

To simplify the rowsize for this project, we are requiring the user to enter a width in pixels that is an integer multiple of 4.

Therefore, rowsize will simply be (number of pixels per row)*(3).
Once you have computed rowsize, malloc( ) a 1-D vector of unsigned chars to hold the RGB values and any zero-padding bytes.
6) Loop through the number of rows of the image as specified by the user.
During each iteration of the loop:
(A) use the rand() function to generate a new set of random BGR values 0-255 for the image row pixels, storing them in the malloc'd 1-D unsigned char array of dimension rowsize.
Make sure that you seed the rand() function using the srand() function with the argument of srand( ) being the time-of-day obtained from your computer using the following syntax. But seed rand( ) using srand( ) only ONE time in your program.
The statement is
srand((unsigned int)time(NULL));
You will need to include the time header file
#include<time.h>
in order to use the function time(NULL).
Recall that ansi function rand( ) generates values between 0 and RAND_MAX, but you want to convert each random number into a number in the range 0 to 255 to be used as BGR values.
(B) Write the random value for the current image row to the BMP file using a single fwrite( ). The 1-D vector that you write will be the one you declared in Step 5) above. Generate a new random vector for each row of the image.
7) close the BMP file and free the malloc'd memory.

If your program is written correctly, after you run your program, you can use your mouse to double-click on your output file and your computer's default application for displaying BMP images will display a pseudo-random color image.