Introduction to Computer Science
HW #1
Due: 2014/03/12

Homework Rules:

Hand-written homework can be handed in in class. Otherwise, you may contact the TA in advance and then bring the hardcopy to the TA in BL421.

As for the programming part, you need to upload it to CEIBA before the deadline (2014/03/13 3am). The file you upload must be a **.zip** file that contains the following files:

README.txt

HW01_b02901XXX (a folder that contains all .cpp & .h as required),

1. Do not submit executable files (.exe) or objective files (.o, .obj). Files with names in wrong format will not be graded. You must **remove any system calls**, such as system ("pause"), in your code if you use it.
2. In README.txt, you need to describe which compiler you used in this homework and how to compile it (if it is in a "project" form).
3. In your .cpp files, we suggest you write comments as detailed as you can. If your code does not work properly, code with comments earns you more partial credits.

Chapter 1 Review Problems (50%)

Problems 3(a), 31, 40, 44, 48

Programming Problem (50%)

We have learned lots of data storage. Don't you ever want to know how exactly images are stored in our computer? Let's try the easiest one: bmp format. In this problem, you are going to write a **BMPImg class** that can:

(1) Load a bmp file. (**In simple format**, no need to deal with arbitrarily case)

(2) Do a simple color transform: transfer the R,G,B color into gray-scale.
(But still store in R,G,B channel, we will talk about it later.)

(3) Store it as another bitmap picture. You may check it by any bmp reader.

Introduction to Computer Science
HW #1
Due: 2014/03/12

How to start? (File format)

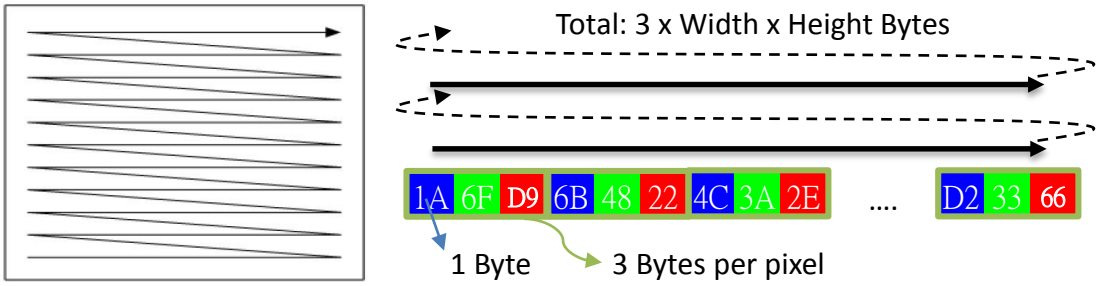
Bitmap files are composed of 2 parts: header and content (bitmap data).
The header stores a table that describes information about this picture. In this homework, we only consider the most common case as follows:

Shift	*Name*	*Size*	*Notes*
0x00	Identifier (ID)	2	Always be "BM" (char)
0x02	File Size	4	Unit: byte
0x06	Reserved	4	0
0x0A	Bitmap Data Offset	4	int(54) **in our case**
0x0E	Bitmap Header Size	4	int(54) **in our case**
0x12	Width	4	Unit: pixel
0x16	Height	4	Unit: pixel
0x1A	Planes	2	1
0x1C	Bits Per Pixel	2	24 for RGB[8,8,8] (this problem) 16 for RGB[5,5,5] (for bonus)
0x1E	Compression	4	0 **in our case** (no compression)
0x22	Bitmap Data Size	4	Unit: bytes
0x26	H-Resolution	4	Keep it untouched
0x2A	V-Resolution	4	Keep it untouched
0x2E	Used Colors	4	0 **in our case**
0x32	Important Colors	4	0 **in our case**

For more detail, you can refer to: http://crazycat1130.pixnet.net/blog/post/1345538
If you want to do it byte-by-byte yourself, be careful of the Little Endian problem.
[Hint]: You don't need to worry about it if you read/write as int/short directly.

Introduction to Computer Science
HW #1
Due: 2014/03/12

As for the content, it depends on the "Bits Per Pixel" and "Compression" to determine the format. **This problem sticks with RGB24 and no-compression.** That means the color data would be store like:



Gray scale:

You must follow this rule to get the gray scale result:

$$Y = 0.299\ R + 0.587\ G + 0.114\ B$$

A little bit inaccuracy is OK, we won't be picky. After calculating the Y value, **store it into both R, G, and B** channels (still RGB24 format).

Hint:

Before you start, you can take a look at the code we provided. Maybe it can inspire you how to finish this problem. You are not asked to follow it strictly, but you need to follow the coding rules below:

(1) A class named as BMPImg. TA will use it for grading.

(2) There must be these member functions as the interface:

```
bool/void loadPic (const string& /const char* picPath); //Load bmp
bool/void RGBtoY  (); //calc Y and store back to RGB
bool/void storePic(const string& /const char* outPath); //Store
```

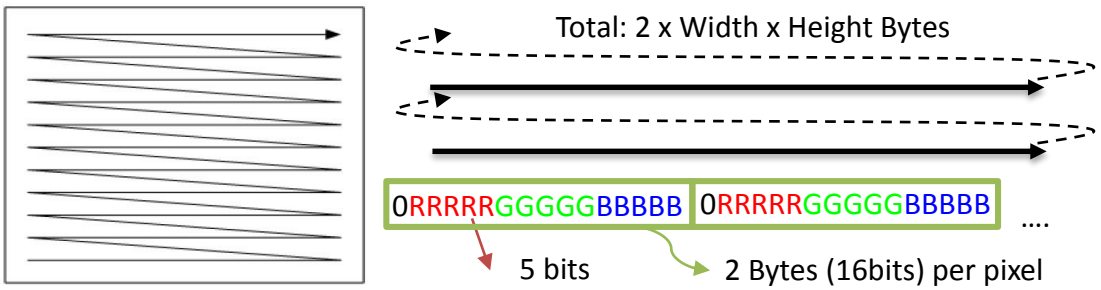
(3) TA will test your code in a way like this:

```
#include "BMPImg.h"
int main(){
    BMPImg img;
    img.loadPic("liver.bmp");
    img.storePic("result1.bmp");
    img.RGBtoY();
    img.storePic("result2.bmp");
    return 0;
}
```

Introduction to Computer Science
HW #1
Due: 2014/03/12

Bonus (5%)

Write a function, bpp24to16, to change the image's storing format "Bits Per Pixel" from 24 to 16. That means you not only need to change some header flags, but also have to do something the data content. There is the required format for RGB555 (16 bits):



TA will test your code in a way like this:

```
#include "BMPImg.h"
int main(){
    BMPImg img;
    img.loadPic("liver.bmp");
    img.storePic("result3.bmp");
    img.bpp24to16();
    img.storePic("result4.bmp");
    return 0;
}
```

If you meet the bonus requirements, write "I finish the bonus." in Readme.txt to let the TA know.

[Hint] after bpp24to16, the header would be: (for liver case)

```
Identifier:BM
FileSize:28854
Reserved:0
BitmapDataOffset:54
BitmapHeaderSize:40
Width:120
Height:120
Planes:1
BitsPerPixel:16
Compression:0
BitmapDataSize:28800
H_Resolution:3780
V_Resolution:3780
UsedColors:0
ImportantColors:0
```