Introduction to Computer Science
HW #1
Due: 2019/03/12

Homework Rules:

Plagiarism is not allowed!!!! If plagiarism is found, you will not be graded.

Hand-written homework can be handed in **before lecture starts**. Otherwise, you may contact the TA in advance and then bring the hardcopy to the TA in MD-631 (please send e-mail in advance).

As for the programming part, you need to upload it to CEIBA before the deadline (2019/03/12 11:59 p.m.). You can choose to program in C++ or in Python. The file you upload must be a **.zip** file that contains the following files:

README.txt
HW01_b06901XXX (a folder that contains all .cpp & .h (or .py) as required),

If you program in C/C++:

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 any.
2. In README.txt, you need to describe which compiler you used in this homework and how to compile or execute 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.

If you program in Python:

1. Please make sure that your program can run on **Python 3.5**.
2. Do not submit .pyc files. Files with names in **wrong format will not be graded**.
3. In your .py 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.
4. **All libraries which contains any functions of image processing (e.g. Python Image Library) are prohibited**. If you are not sure whether a library can be used or not, please contact TA.

Introduction to Computer Science
HW #1
Due: 2019/03/12

Chapter 1 Review Problems (40%)

Problems 3a, 3b, 31, 34, 40, 44, 49, 54.

Programming Problem (60%)

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 cases)
(2) Do a counterclockwise rotation by 90 degrees.
(3) Store it as another bitmap picture. You may check it by any bmp reader.

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 (bytes)*	*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(40) **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] (**in our case**) 16 for RGB[5,5,5]
0x1E	Compression	4	0 **in our case** (no compression)
0x22	Bitmap Data Size	4	Unit: bytes
0x26	H-Resolution	4	Keep it untouched

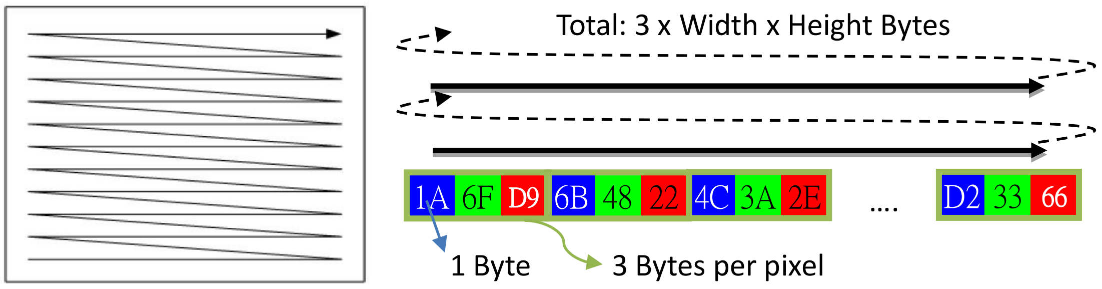
Introduction to Computer Science
HW #1
Due: 2019/03/12

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

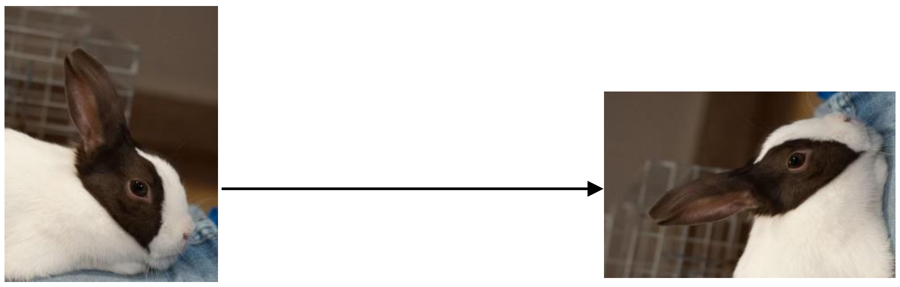
Hint: You don't need to worry about it if you read/write as int/short directly.

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 stored like this:


Total: 3 x Width x Height Bytes
1A 6F D9 6B 48 22 4C 3A 2E D2 33 66
1 Byte
3 Bytes per pixel

Rotation:

In this homework, you are asked to **turn a colored image counterclockwise by 90 degrees**.



Hint:

- Before you start, you can have a look at the code we provided. Maybe it can inspire you how to finish this problem easily. You are not requested to follow it strictly, but you need to obey the homework correcting rules below.
- For binary file read/write, here are excellent tutorials:
  - In C++:
    http://www.cplusplus.com/doc/tutorial/files/
    Remember to use the "ios::binary" flag.

- In Python:
  https://www.tutorialspoint.com/python/python_files_io.htm
  Remember to use the “b“ mode.
  Furthermore, you may use “struct” for the data type conversion.
  https://docs.python.org/3/library/struct.html

Important rules:

You **MUST** follow these coding rules:

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

(2) There must be these member functions as interfaces:

In C++:

```
bool/void loadPic (string/char* picPath); //Loading bmp file
bool/void rotate (); //calculate corresponding pixel
bool/void storePic(string/char* outPath); //Store bmp file
```

In Python:

```
def loadPic (self, picPath):
    #Loading bmp file
def rotate (self):
    #calculate corresponding pixel
def storePic(outputPath):
    #Store bmp file
```

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

In C++:

```
#include "BMPImg.h"
#include <string>
int main(int argc, char* argv[]){
    BMPImg imgrabbit("img/rabbit.bmp");
    imgrabbit.rotate()
    imgrabbit.storePic("output/rabbit_ans.bmp");
    return 0;
}
```

Introduction to Computer Science
HW #1
Due: 2019/03/12

In Python:

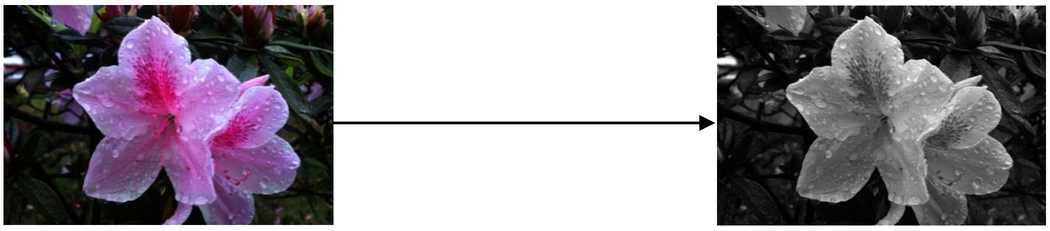
```
import BMPImg;
def main:
    bmpimg = BMPImg.BMPImg();
    img.loadPic("img/rabbit.bmp");
    img.rotate();
    img.storePic("output/rabbit_ans.bmp");
```

Bonus (5%): Prewiit Edge Detector

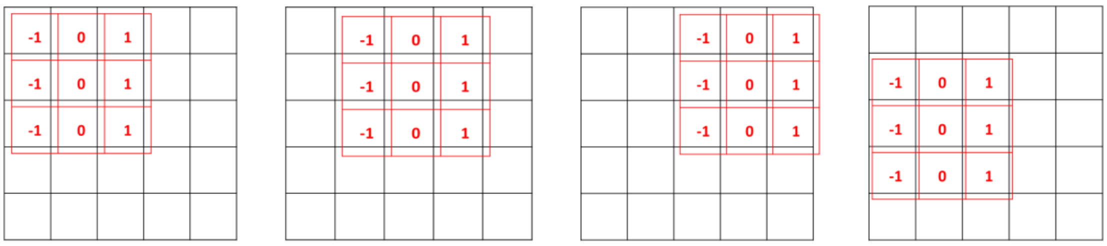
In this homework, we first change the picture to gray scale and then apply Prewiit filter on it. The output format should within 3 channels.
Here, we are going do something special on our images. First, we have to convert the image to grayscale.
Gray-scale value is calculated via

$$Y = 0.3R + 0.59G + 0.11B$$

For the decimal part, you need to round down the number, and negative value should be treated as zero. After calculating the Y value, store it into R, G, and B channels (still RGB24 format).



Next, there is an easy but useful function to detect edges in a picture by Prewiit filter. We compute a 3x3 filter, containing 9 pixels in total. Then we calculate the value G as follows and assign it to the center of the filter.

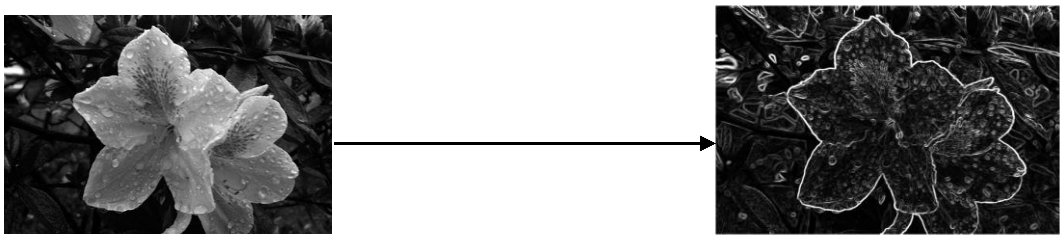


Introduction to Computer Science
HW #1
Due: 2019/03/12

$$\begin{matrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{matrix} \quad \begin{cases} G_x = -x_1 - x_4 - x_7 + x_3 + x_6 + x_9 \\ G_y = -x_1 - x_2 - x_3 + x_7 + x_8 + x_9 \end{cases} \quad G = \sqrt{G_x^2 + G_y^2}$$

13	7	9
5	3	1
12	1	4

The value is 20 in the above filter. Then we apply this 3x3 filter across the whole picture, resulting a smaller picture (orig_width-2)*(orig_height-2), and that's it. To make this homework simple, you won't change the dimensions; instead, simply set the borders to 255 (the 1st and the last rows, columns).



TA will test your code this way like:

In C++:

```
#include "BMPImg.h"
int main(){
    BMPImg imgflower("./img/flower.bmp");
    imgflower.PrewittFilter();
    imgflower.storePic("./output/flower_ans.bmp");
    return 0;
}
```

In Python:

```
Import BMPImg
def main:
    bmpimg = BMPImg.BMPImg();
    img.loadPic("./img/flower.bmp");
    img.PrewiitFilter();
    img.storePic("./output/flower_ans.bmp");
```

Introduction to Computer Science
HW #1
Due: 2019/03/12

If you meet the bonus requirements, write “I finished the bonus part.” in the readme file to let TA know.