READ ME

I. Description

This program is a image tone mapping tool which can be used to read a high dynamic range image and output a tone mapped image with a proper dynamic range. There are 3 modes in this program, including Simple Tone Mapping, Tone Mapping with Convolution and Tone Mapping with Convolution using a bilateral filter.
The program includes 4 files, which are 'my.h', 'tonemap.cpp', ‘Makefile’, and ‘README.pdf'. The folder also has 6 images, ‘simpleocean.png’, ‘simplesmall.png’, ‘cvlvocean.png’, ‘cvlvsmall.png’, ‘bilaterocean.png’, ‘bilatersmall.png’.
To compile on linux, use the Makefile that is included in the folder of this program.

Author: Siqi Wu
Email: siqiw@clemson.edu
Data: 10/27/2015

2. Data Structure And Methods

1). Object-Oriented Data Structure
In this program, a class is built for image. The class includes the basic information and some manipulation functions of the image.

Class Name: MyImage
Basic Information:

```
char *filename;
int    height;
int    width;
int    channel;
```

```
int    ochannel;
int    oheight;
int    owidth;
float* data;
float* originaldata;
```

Manipulation Functions:

```
void       SetFilename(char *newname){ filename = newname;}
void       SetOriginalData ();
void       ImageRead();
void       ImageWrite(char *);
float*     ImageGetData(){ return data; }
float*     GetOriginalData(){ return originaldata; }
int        GetHeight(){ return height; }
int        GetOHeight(){ return height; }
int        GetWidth(){ return width; }
int        GetOWidth(){ return owidth; }
int        GetChannel(){ return channel; }
int        GetOChannel(){return ochannel;}
char*      GetFilename(){ return filename; }
void       SetHeight( int seth ){ height = seth; }
void       SetWidth( int setw ){ width = setw; }
void       SetData( float* newdata){data = newdata;}
void       SetOHeight( int seth ){ oheight = seth; }
void       SetOWidth( int setw ){ owidth = setw; }
void       SetOChannel( int setc ){ ochannel = setc; }
void       SetChannel( int setc ){ channel = setc; }
void       ImageDisplay();
float*     simpleTM( float );
float*     convolveTM( float**, int, float, int )
float*     convolve( float* img, int hei, int ch, int number, float** kernel, int tag)
```

2). Method used in Bilateral Filter

The method used to implement Bilateral Filter is to multiply a weight of w=exp(-clamp(d^2)). When the '-b' mode is set, for each pixel in the kernel window, calculate w = exp((-1)*(val - img[r*wid+c])*(val - img[r*wid+c])), where img[r*wid +c] is the center of window. val is the value of this pixel. When it falls outside of the pixel, it is set to zero. Then the convolution process becomes cvlv= cvlv + val * k * w, which means to sum up all the results of val * k * w around the window. k is the value of the kernel at corresponding position. After that, the scale factor here becomes the positive values of the kernel times w, to make sure they have the same range as before.
Result of the best Bilateral Filtering is submitted as bilateral.png.

3. Introduction of Functions

1. Functions in file 'my.h'

```
void    SetFilename(char *newname){ filename = newname;}
void    SetOriginalData ();
void    ImageRead();
void    ImageWrite(char *);
float*  ImageGetData(){ return data; }
float*  GetOriginalData(){ return originaldata; }
int     GetHeight(){ return height; }
int     GetOHeight(){ return height; }
int     GetWidth(){ return width; }
int     GetOWidth(){ return owidth; }
int     GetChannel(){ return channel; }
int     GetOChannel(){return ochannel;}
char*   GetFilename(){ return filename; }
void    SetHeight( int seth ){ height = seth; }
void    SetWidth( int setw ){ width = setw; }
void    SetData( float* newdata){data = newdata;}
void    SetOHeight( int seth ){ oheight = seth; }
void    SetOWidth( int setw ){ owidth = setw; }
```

```
void       SetOChannel( int setc ){ ochannel = setc; }
void       SetChannel( int setc ){ channel = setc; }
void       ImageDisplay();
float*     simpleTM( float );
float*     convolveTM( float**, int, float, int )
float*     convolve( float* img, int hei, int ch, int number, float** kernel, int tag)
```

2. Function in file 'oiioview.cpp'

```
void    RenderScene()
void    handleKey( unsigned char key, int x, int y )
```

4. User Instructions

1). Command Line Arguments

This program can be used in 3 modes. The first one is Simple Tone Mapping. The second one is Tone Mapping with Convolution. And the third one is Tone Mapping with Convolution using Bilateral Filter. For each mode, user can write the displaying image if a write file name is specified.

Usage: ./tonemap [flag][filename][gamma]([kernelname])([writename]).
Mode1: ./tonemap -g [filename][gamma]([writename]).
Mode2: ./tonemap -c [filename][gamma][kernelname]([writename]).
Mode3: ./tonemap -b [filename][gamma][kernelname]([writename]).

2). Keyboard Manipulation

I. Press 's' or 'S' to switch between the original and tone mapped image.
II. Press 'w' or 'W' to write the displaying image as the file name of the second argument.
IV.Press 'q' or 'Q' to quit the program.

5. Experiment Result

1. Simple Tone Mapping

1). Ocean.exr

For the Ocean.exr picture, the best Simple Tone Mapping result is when gamma = 0.35. The result is stored as ‘simpleocean.png’.

2) smalldesignCenter.hdr

For the image smalldesignCenter.hdr, the best Simple Tone Mapping result is when gamma = 0.08. The result is stored as ‘simplesmall.png’.

2. Tone Mapping with Convolution

1). Ocean.exr

For the Ocean.exr picture, the best Tone Mapping with Convolution result is when gamma = 0.35 using convolution kernel ‘bell9.filt’. The result is stored as ‘cvlvocean.png’. Larger scale of kernel may have better result, but has more computation complexity.

2) smalldesignCenter.hdr

For the image smalldesignCenter.hdr, the best Tone Mapping with Convolution result is when gamma = 0.08 using convolution kernel ‘bell9.filt’. The result is stored as ‘cvlvsmall.png’. Larger scale of kernel may have better result, but has more computation complexity.

3. Tone Mapping with Convolution with Bilateral Filter

1). Ocean.exr

For the Ocean.exr picture, the best Tone Mapping with Convolution result is when gamma = 0.35 using convolution kernel ‘bell9.filt’. The result is stored as ‘bilaterocean.png’. Larger scale of kernel may have better result, but has more computation complexity.

2) smalldesignCenter.hdr

For the image smalldesignCenter.hdr, the best Tone Mapping with Convolution result is when gamma = 0.07 using convolution kernel ‘bell9.filt’. The result is stored as ‘bilatermall.png’. Larger scale of kernel may have better result, but has more computation complexity.