**Jessie George**

HW 2

**Question 1 (a)**

**Code:**

**import** ij.plugin.PlugIn;

**import** ij.ImagePlus;

**import** ij.process.FloatProcessor;

**import** ij.process.ImageProcessor;

**import** ij.ImageStack;

**public** **class** create\_Image **implements** PlugIn {

ImageProcessor makeRandomImage(**int** w, **int** h) {

**int** alpha, red, green, blue;

**int**[] p = **new** **int**[w \* h];

**for** (**int** j = 0; j < h; j++)

{

**for** (**int** i = 0; i < w; i++)

{

//range [0, 255]

alpha = (**int**)(Math.*random*()\*256);

red = (**int**)(Math.*random*()\*256);

green = (**int**)(Math.*random*()\*256);

blue = (**int**)(Math.*random*()\*256);

p[i + w \*j] = (alpha<<24) | (red<<16) | (green<<8) | blue;

}

}

**return** **new** FloatProcessor(w, h, p);

}

**public** **void** run(String arg)

{

**int** w = 320, h = 240;

ImageStack stack = **new** ImageStack(w, h);

**for** (**int** i = 0; i < 180; i++)

{

stack.addSlice("", makeRandomImage(w, h));

}

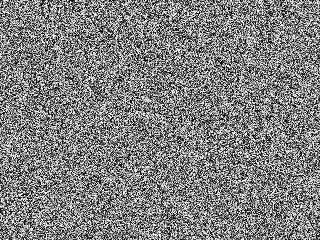
ImagePlus image = **new** ImagePlus("stack", stack);

image.show();

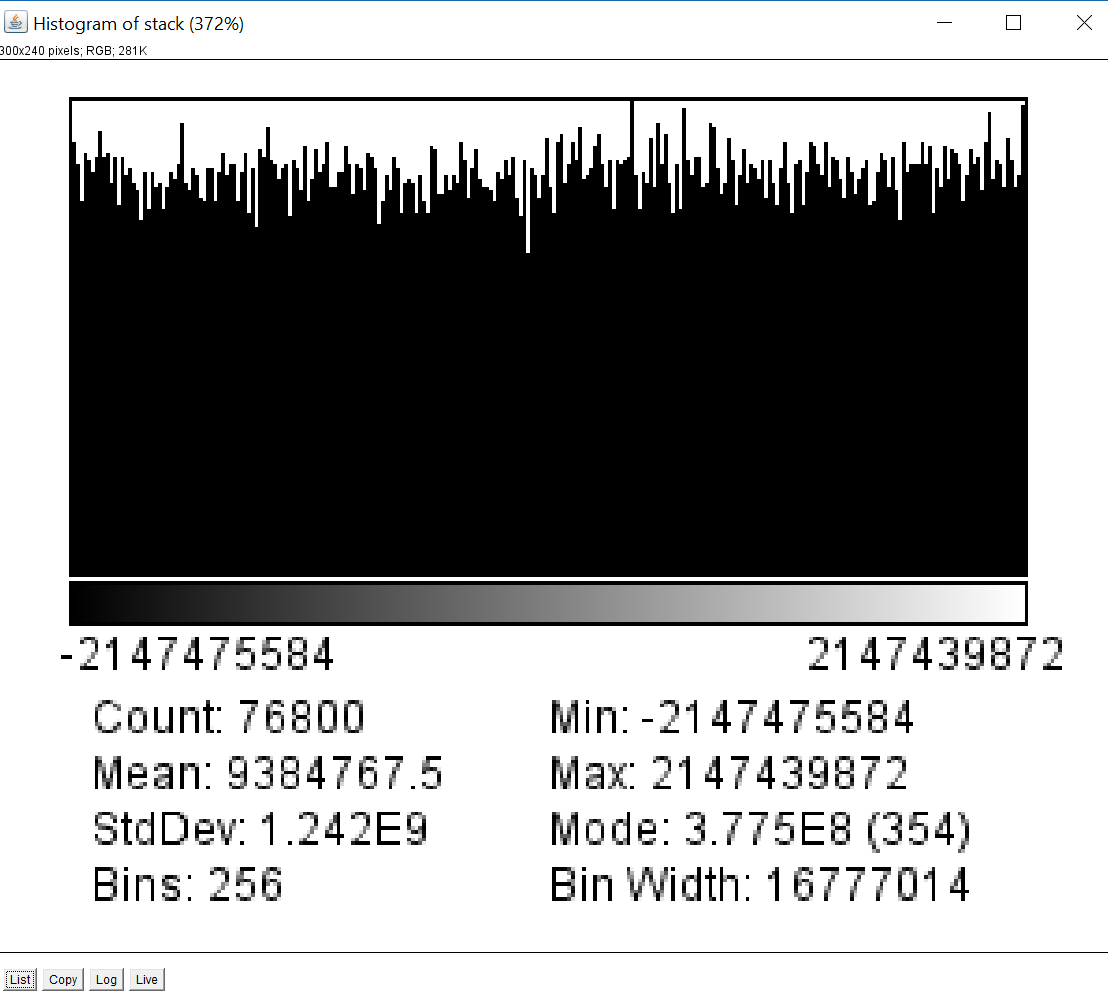
}

}

**Image 1(a):**

****

**Histogram 1(a):**

****

From the above histogram, we see that the pixel values are uniformly distributed. (The histogram is box shaped.)

**Question 1 (b)**

**Code:**

**import** ij.plugin.PlugIn;

**import** ij.ImagePlus;

**import** ij.process.FloatProcessor;

**import** ij.process.ImageProcessor;

**import** ij.ImageStack;

**import** java.util.Random;

**public** **class** create\_GaussianImage **implements** PlugIn {

ImageProcessor makeGausImage(**int** w, **int** h) {

Random r = **new** Random();

**double**[] p = **new** **double**[w \* h];

**for** (**int** j = 0; j < h; j++)

{

**for** (**int** i = 0; i < w; i++)

{

p[i + w \*j] = r.nextGaussian()\*50 + 150;

}

}

**return** **new** FloatProcessor(w, h, p);

}

**public** **void** run(String arg)

{

**int** w = 320, h = 240;

ImageStack stack = **new** ImageStack(w, h);

**for** (**int** i = 0; i < 180; i++)

{

stack.addSlice("", makeGausImage(w, h));

}

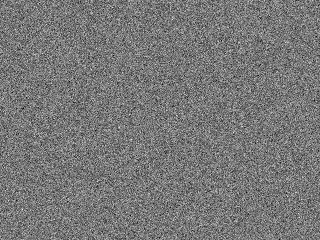
ImagePlus image = **new** ImagePlus("Question 1 b", stack);

image.show();

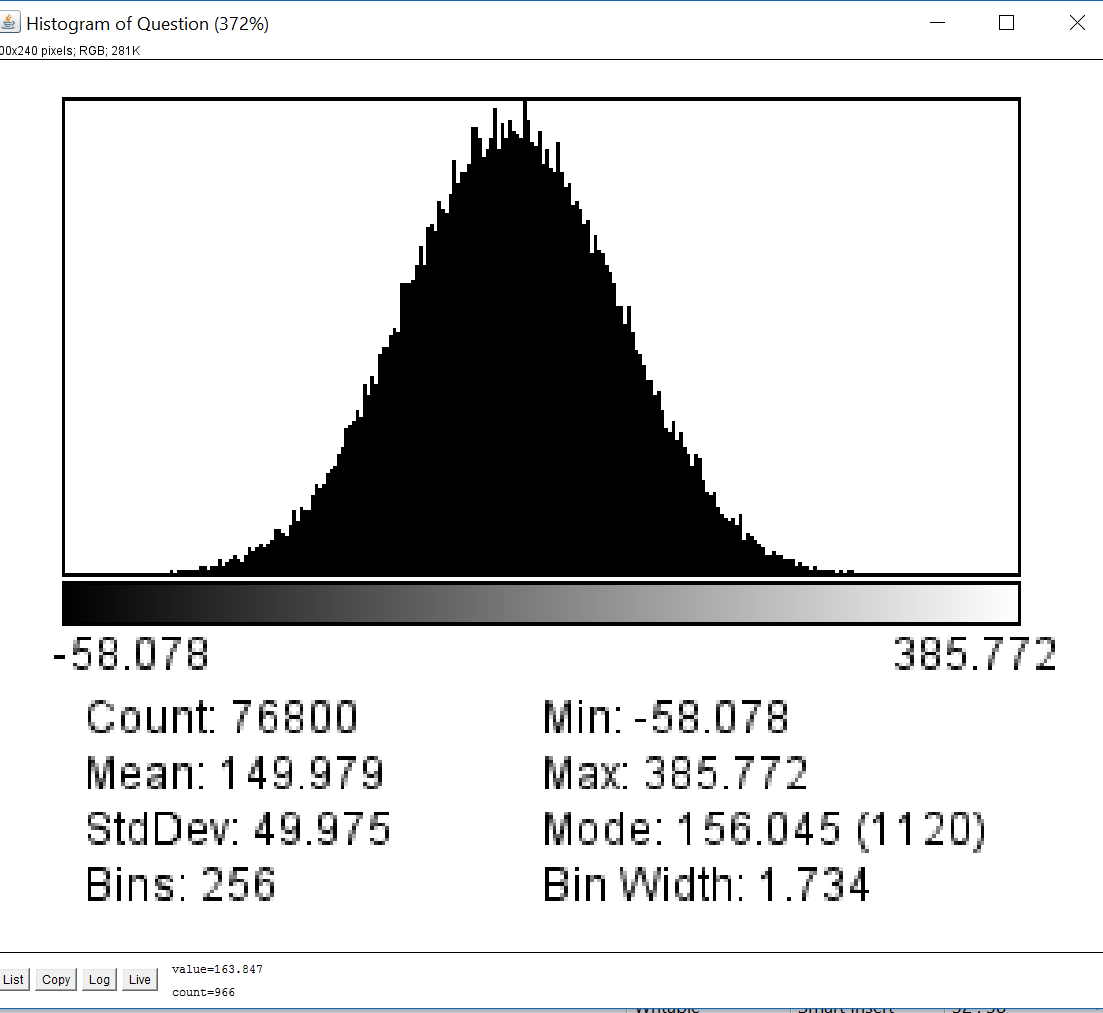
}

}

**Image 1(b):**

****

**Histogram 1(b):**

****

**Question 2**

**Code**

**import** ij.IJ;

**import** ij.ImagePlus;

**import** ij.plugin.PlugIn;

**import** ij.process.ImageProcessor;

**public** **class** count\_ColorPixels **implements** PlugIn{

**public** **void** run(String args)

{

ImagePlus im = IJ.*getImage*();

ImageProcessor ip = im.getProcessor();

**int** w = ip.getWidth();

**int** h = ip.getHeight();

/\*

count the number of pixels with pure red

(255,0,0), pure green (0,255,0), pure blue (0,0,255), white

(255,255,255), and black (0,0,0)

\*/

**int** numRed, numGreen, numBlue, numWhite, numBlack;

numRed = numGreen = numBlue = numWhite = numBlack = 0;

**for**(**int** u = 0;u<w;u++)

{

**for**(**int** v = 0;v<h;v++)

{

**int**[] rgb = **new** **int**[3];

ip.getPixel(u,v,rgb);

**int** r = rgb[0];

**int** g = rgb[1];

**int** b = rgb[2];

**if**(r==255 && g==0 && b==0)

numRed++;

**else** **if**(r==0 && g==255 && b==0)

numGreen++;

**else** **if**(r==0 && g==0 && b==255)

numBlue++;

**else** **if**(r==255 && g==255 && b==255)

numWhite++;

**else** **if**(r==0 && g==0 && b==0)

numBlack++;

}

}

IJ.*log*("Number of pure Red pixels = "+numRed);

IJ.*log*("Number of pure Green pixels = "+numGreen);

IJ.*log*("Number of pure Blue pixels = "+numBlue);

IJ.*log*("Number of pure White pixels = "+numWhite);

IJ.*log*("Number of pure Black pixels = "+numBlack);

}

}

**Selfie**

****

**Output for selfie**

Number of pure Red pixels = 0

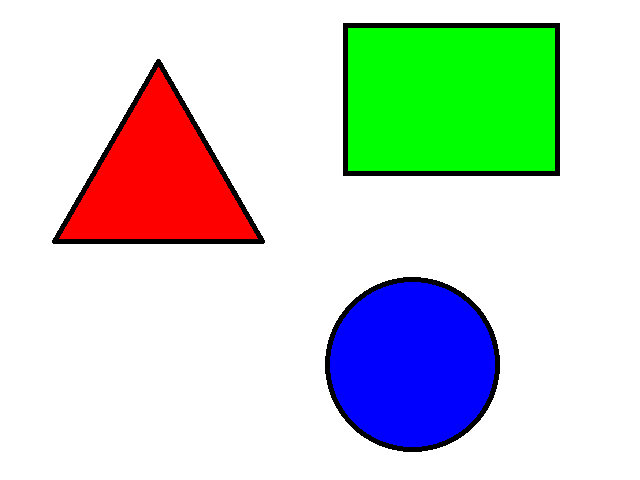
Number of pure Green pixels = 0

Number of pure Blue pixels = 0

Number of pure White pixels = 0

Number of pure Black pixels = 0

**Shapes**

****

**Output for Shapes**

Number of pure Red pixels = 17197

Number of pure Green pixels = 29601

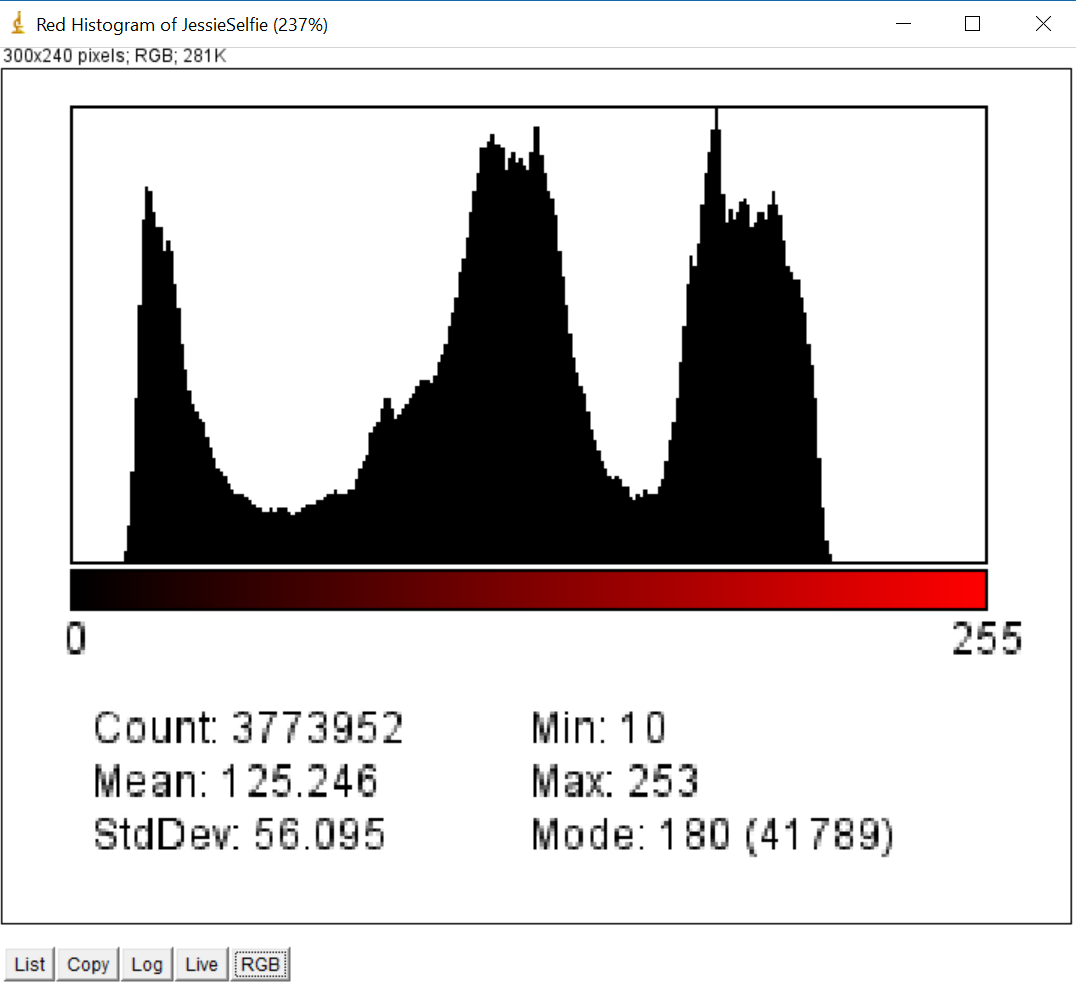
Number of pure Blue pixels = 21377

Number of pure White pixels = 229662

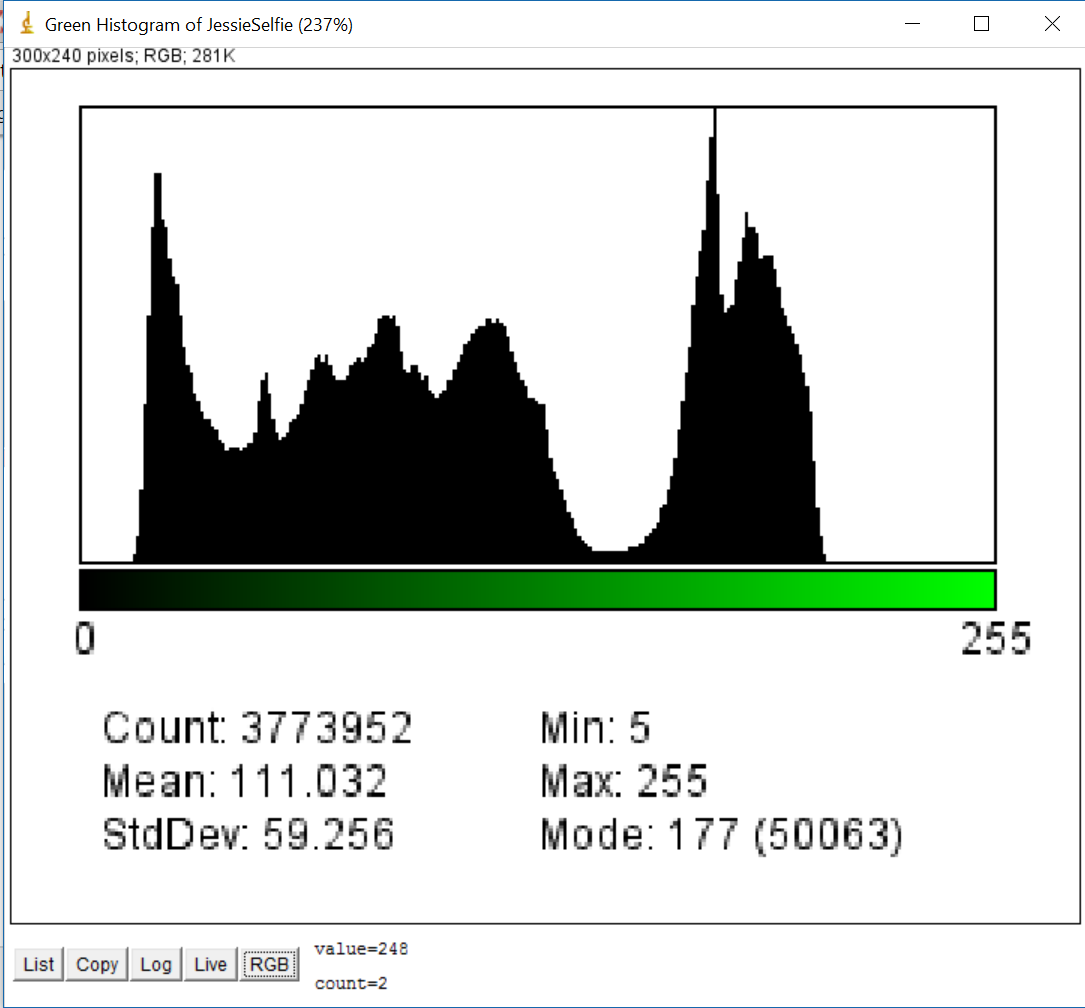
Number of pure Black pixels = 9363

**Question 3 (a)**

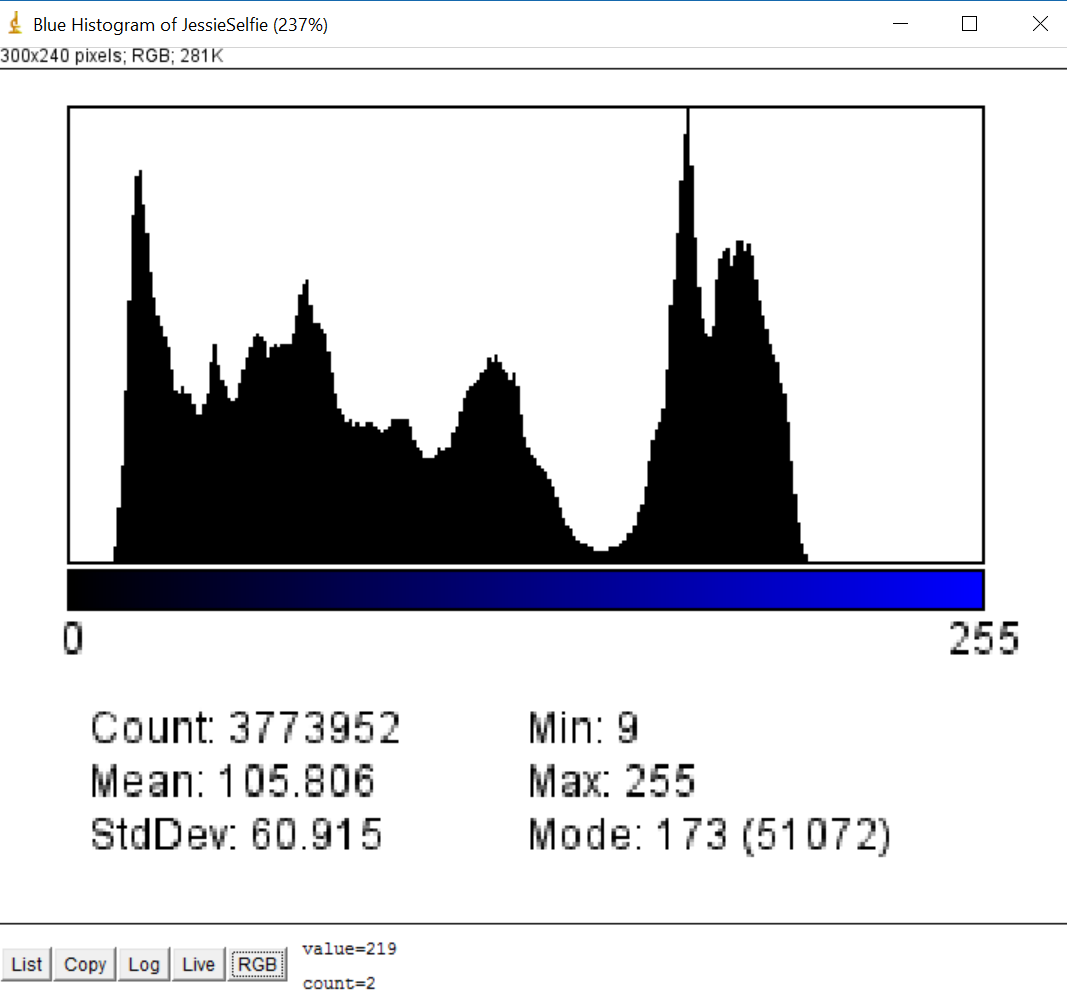
**Red Component histogram**

****

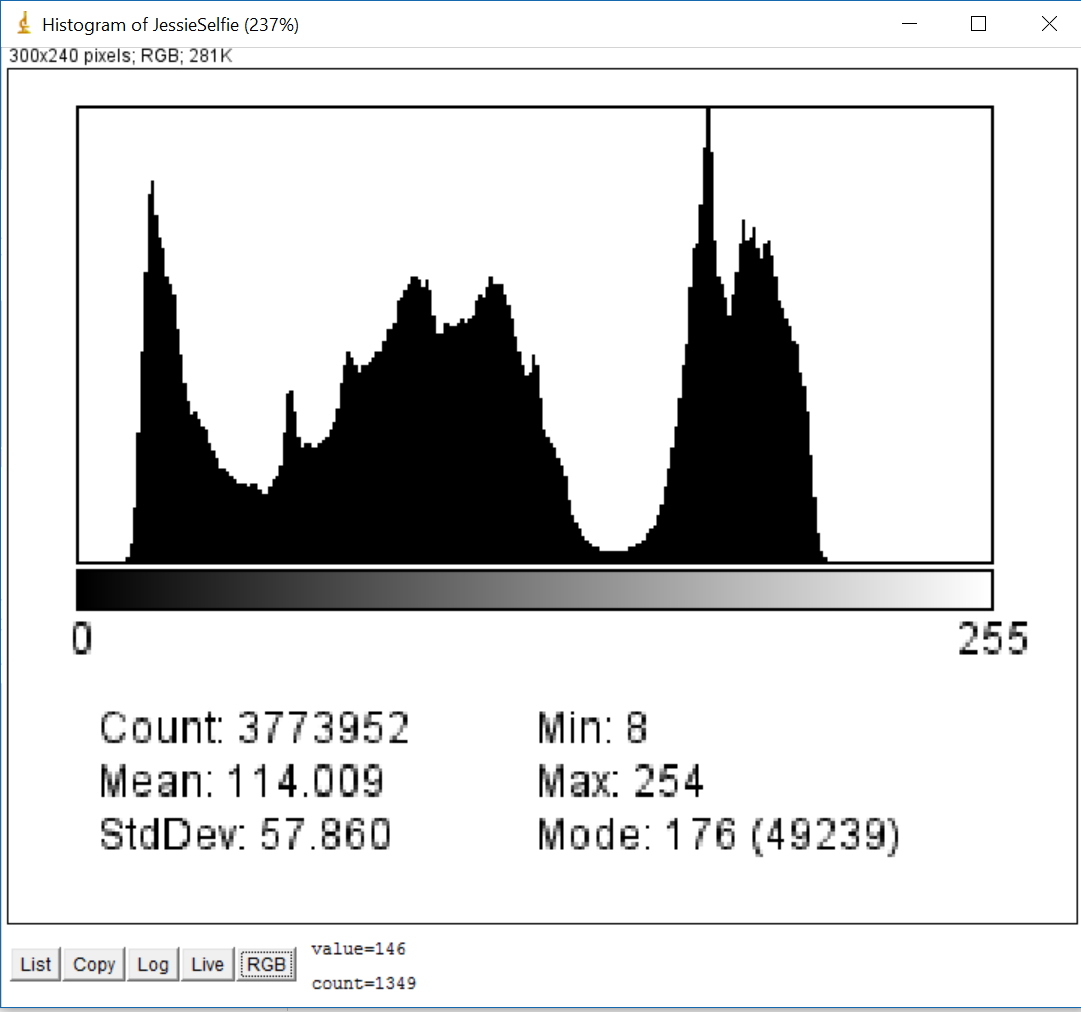
**Green Component histogram**

****

**Blue Component histogram**

****

**Luminance histogram**

****

**Question 3 (b)**

Image has spike because of jpg compression effect.

**Question 3 (c)**

**Code**

import ij.IJ;

import ij.ImagePlus;

import ij.plugin.PlugIn;

import ij.process.ImageProcessor;

public class convert\_Grayscale implements PlugIn{

public void applyGammaFunction(ImageProcessor ip, double GAMMA)

{

ImageProcessor ipNew = ip.duplicate();

// works for 8-bit images only

int K = 256;

int aMax = K - 1;

// create and fill the lookup table:

int[] Fgc = new int[K];

for (int a = 0; a < K; a++) {

double aa = (double) a / aMax; // scale to [0, 1]

double bb = Math.pow(aa, GAMMA); // power function

// scale back to [0, 255]:

int b = (int) Math.round(bb \* aMax);

Fgc[a] = b;

}

ipNew.applyTable(Fgc); // modify the image

ImagePlus imPls = new ImagePlus("Gamma"+GAMMA,ipNew);

imPls.show();

}

public void run(String args)

{

ImagePlus im = IJ.getImage();

ImageProcessor ip = im.getProcessor();

ImageProcessor ipNew = ip.duplicate();

int w = ip.getWidth();

int h = ip.getHeight();

for(int u = 0;u<w;u++)

{

for(int v = 0;v<h;v++)

{

int[] rgb = new int[3];

ip.getPixel(u,v,rgb);

int r = rgb[0];

int g = rgb[1];

int b = rgb[2];

int gray = (r+g+b)/3;

int grayVal = (gray<<16) | (gray<<8) | gray;

ipNew.set(u,v,grayVal);

}

}

ImagePlus imPls = new ImagePlus("Before Gamma",ipNew);

imPls.show();

applyGammaFunction(ipNew, 0.25);

applyGammaFunction(ipNew, 0.5);

applyGammaFunction(ipNew, 2.0);

applyGammaFunction(ipNew, 4.0);

}

}

**Grayscale Image Before applying Gamma Function (i.e. just grayscale using average method)**

****

**After applying Gamma Function**

**top row left to right is Gamma 0.25 and Gamma 0.5**

**bottom row left to right is Gamma 2.0 and Gamma 4.0**

** **

** **

**Question 3 (d)**

The image with Gamma=0.5 show the most faithful visualization because the printer also performs a gamma transformation. (So if we add the two gamma transformations, it should be close to gamma=1 which is the original)

**Question 4 (a)**

**Code**

**import** java.util.Random;

**import** ij.IJ;

**import** ij.ImagePlus;

**import** ij.plugin.PlugIn;

**import** ij.process.FloatProcessor;

**import** ij.process.ImageProcessor;

**public** **class** add\_Noise **implements** PlugIn{

**public** **void** makeNoisy(ImageProcessor ip, **int** mean, **int** stdDev)

{

ImageProcessor ipNew = ip.duplicate();

**int** w = ipNew.getWidth();

**int** h = ipNew.getHeight();

Random r = **new** Random();

**for** (**int** j = 0; j < h; j++)

{

**for** (**int** i = 0; i < w; i++)

{

**int** p = ip.getPixel(i,j) + (**int**)(r.nextGaussian()\*stdDev + mean);

ipNew.set(i,j,p);

}

}

ImagePlus imPls = **new** ImagePlus("stdDev"+stdDev,ipNew);

imPls.show();

}

**public** **void** run(String args)

{

ImagePlus im = IJ.*getImage*();

ImageProcessor ip = im.getProcessor();

makeNoisy(ip, 0, 5);

makeNoisy(ip, 0, 15);

makeNoisy(ip, 0, 25);

}

}

**Original Grayscale Image (See next page)**

****

**Standard Deviation = 5 (see next page)**

****

**Standard Deviation = 15 (see next page)**

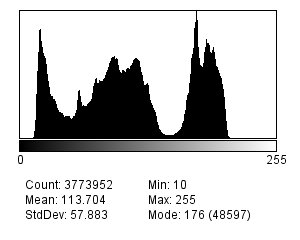
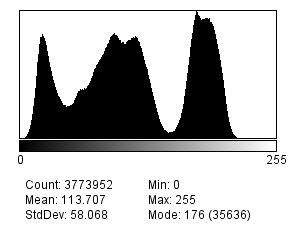
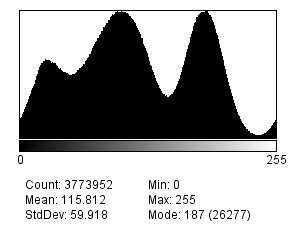
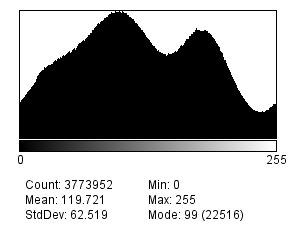
****

**Standard Deviation = 25 (see next page)**

****

**Top row Left to Right – Histogram for Original grayscale Image, Histogram for standard deviation = 5**

**Bottom row Left to Right –Histogram for standard deviation = 15,Histogram for standard deviation=25**

**** ****  

By adding noise, the histogram curve gets smoother and loses detail i.e. the original histogram has many peaks but the last histogram has few peaks.

**Question 4 (b)**

**Code:**

**import** ij.plugin.filter.GaussianBlur;

**import** ij.ImagePlus;

**import** ij.plugin.filter.PlugInFilter;

**import** ij.process.ImageProcessor;

**public** **class** Gaussian\_Filter **implements** PlugInFilter {

**public** **int** setup(String arg, ImagePlus im)

{

**return** ***DOES\_8G***; //accepts 8-bits gray scale

}

**public** **void** run(ImageProcessor ip)

{

GaussianBlur gb = **new** GaussianBlur();

gb.blurGaussian(ip, 3); //I changed this to 5 and 7 respectively

}

}

**3x3 Gaussian filter for noisy image 1 (standard deviation = 5)**

****

**3x3 Gaussian filter for noisy image 2 (standard deviation = 15)**

****

**3x3 Gaussian filter for noisy image 3 (standard deviation = 25)**

****

**5x5 Gaussian filter for noisy image 1 (standard deviation = 5)**

****

**5x5 Gaussian filter for noisy image 2 (standard deviation = 15)**

****

**5x5 Gaussian filter for noisy image 3 (standard deviation = 25)**

****

**7x7 Gaussian filter for noisy image 1 (standard deviation = 5)**

****

**7x7 Gaussian filter for noisy image 2 (standard deviation = 15)**

****

**7x7 Gaussian filter for noisy image 3 (standard deviation = 25) **

**Question 5 (a)**

**Question 5 (b)**