```
// image.h

#pragma once

#include <string.h>

// image.h
typedef struct image {
	int w;
	int h;
	int c;
	unsigned char* data;
} image;

image load_image(const char* filename);
image make_image(int w, int h, int c);
image make_empty_image(int w, int h, int c);
image RGBtoIntensity(image im);

image Intensity2RGB(image im);




// ****************************************************************************
// image.cpp

#include "image.h"

#define STB_IMAGE_IMPLEMENTATION
#include "stb/include/stb_image.h"
```

```
image load_image(const char* filename)
{
	int w, h, c; // width , height, channel
	int channel = 3;
	//w = width, h = height, c = # 8 - bit components per pixel ...
	unsigned char* data = stbi_load(filename, &w, &h, &c, channel);    // without OpenCV

	if (!data) {
		exit(EXIT_FAILURE);
	}

	image out;
	out.data = data;
	out.h = h;
	out.w = w;
	out.c = c;
	return out;
}//load_image

void Free(image im)
{
	delete[] im.data;
}

image RGBtoIntensity(image im)
{
	image raw;
	raw.data = new unsigned char[im.h * im.w]; // height*weight kadar yer aç
	raw.w = im.w;
	raw.h = im.h;
	raw.c = 1; // intensity-gray level'a çek, tek boyut
	long bufpos = 0;
	long newpos = 0;
	for (int row = 0; row < im.h; row++)
	{
		for (int column = 0; column < im.w; column++)
		{
			newpos = row * im.w + column;
			bufpos = row * im.w * im.c + column * im.c;
			raw.data[newpos] = unsigned char(0.30 * im.data[bufpos] + 0.59 *
im.data[bufpos + 1] + 0.11 * im.data[bufpos + 2]);
		}
	}
	return raw;
}


image Intensity2RGB(image im) {
	image rgb;
	rgb.data = new unsigned char[im.h * im.w * 3]; // R, G, B için 3 kanal
	rgb.w = im.w;
	rgb.h = im.h;
	rgb.c = 3; // RGB formatında çıktı

	long bufpos = 0;
	long newpos = 0;
	for (int row = 0; row < im.h; row++) {
		for (int column = 0; column < im.w; column++) {
			newpos = row * im.w + column;
			bufpos = newpos * 3;
```

```
                        unsigned char intensity = im.data[newpos];

                        rgb.data[bufpos] = intensity / 0.3;       // R kanalına intensity değerini kopyala
                        rgb.data[bufpos + 1] = intensity / 0.59;    // G kanalına intensity değerini
kopyala
                        rgb.data[bufpos + 2] = intensity / 0.11;    // B kanalına intensity değerini
kopyala
                }
        }

        return rgb;
}
```