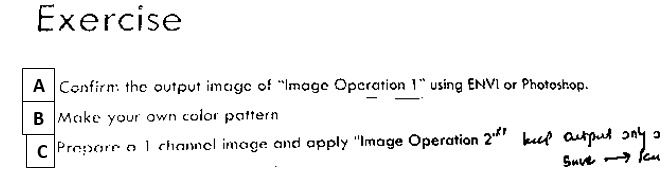
**IMAGE HANDLING AND FILE I/O**

**Objective**

1. Make own color pattern using Input and output image handling knowledge
2. Prepare one channel image using Input and output image handling knowledge



**A and B).**

**Code’s description:**

#include<stdio.h>

#include<stdlib.h>

#include<math.h> **// for Math functions**

#include<malloc.h> **// for memory allocation**

typedef unsigned char u\_char; **//abbreviation**

typedef unsigned short u\_short; **//abbreviation**

**//define image properties/three variables, width, height and number of channel (3). All w,h and nchanel are integer.**

#define W 256

#define H 256

#define NCHAN 3

**/\*Function statement for setup image, write image, allocate memory for image and**

**clear memory after process was done. \*\*\*img is, image pointer, which pointer to the pointer.\*/**

void img\_setup(u\_char \*\*\*img, int w, int h, int nchan**); // function for setup image**

int image\_write(char \*fname, u\_char \*\*\*img, int w, int h, int nchn); **// function for create the output image to be just like the input image (same type and size), name=fname**

u\_char \*\*\*img1\_alloc(int w,int h,int nchan); **// function for allocate memory for image**

void img\_free(u\_char \*\*\*img, int h, int nchan); **// function for clear memory after process**

u\_char \*\*img2\_alloc(int w, int h);

void img2\_free(u\_char \*\*img, int h);

**//Main function part**

main()

{

u\_char \*img;

img = img1\_alloc(W,H,NCHAN); **//Allocate memory for storing an image (W\*H\*NCHANNEL= size of image)**

if (img == NULL){

printf("Not enough memory\n"); exit(1); **// if there is no enough memory in PC exist**

}

**//Setup new image, width=256, height=256 and 3 channel**

img\_setup(img, W, H, NCHAN);

**/\*Write output image, name of result is “test.img”, width, height and channel are equal**

**image setup\*/**

image\_write("test.img", img, W, H, NCHAN);

**//Clear memory after writing process is done**

img\_free(img, H, NCHAN);

}

**//Function to setup new image and store to memory**

#define PI 3.14159265

void img\_setup(u\_char \*\*\*img, int w, int h, int nchan)

{

int i, j, ichan;

for (ichan = 0; ichan < nchan; ichan++){

printf("%d\n", ichan);

for (i = 0; i < h; i++){

for (j = 0; j < w; j++){

**/\* The pattern of image depend on this line od code\*/**

img[ichan][i][j] = sin((double)j / (w\*(1 + ichan\*0.5)) \* 2 \* PI\*(10.0\*i / h)) \* 128 + 128;

}

}

}

printf("img\_setup end \n");

}

**//Function to write data in memory to output file**

int image\_write(char \*fname, u\_char \*\*\*img, int w, int h, int nchan)

{

FILE \*fp;

int i, ichan;

**//open file name that we assign in main function**

if ((fp = fopen(fname, "wb")) == NULL){

fprintf(stderr, "image\_write:file open error");

return 1;

}

printf("file open end\n");

**//Write image by each pixel**

for (ichan = 0; ichan < nchan; ichan++)

for (i = 0; i < h; i++)

fwrite(img[ichan][i], w, 1, fp);

fclose(fp);

return 0;

}

**/\*Function to allocate memory for whole image depend on width, height and number of channel\*/**

u\_char \*\*\*img1\_alloc(int w, int h, int nchan)

{

u\_char \*\*\*img;

int ichan, j;

**//Using malloc to allocate memory dynamically**

if ((img = malloc(sizeof(u\_char\*\*)\*nchan)) == NULL){

return NULL;

}

for (ichan = 0; ichan < nchan; ichan++){

img[ichan] = img2\_alloc(w,h);

if (img[ichan] == NULL){

for (j = 0; j < ichan; j++)

img2\_free(img[j],h);

free(img);

return NULL;

}

}

return img;

}

void img\_free(u\_char \*\*\*img, int h, int nchan)

{

int i;

for (i = 0; i < nchan; i++)

img2\_free(img[i], h);

free(img);

}

void img2\_free(u\_char \*\*img, int h)

{

int i;

for (i = 0; i < h; i++)

free(img[i]);

free(img);

}

**/\*Function to allocate memory for each pixel depend on width and height\*/**

u\_char \*\*img2\_alloc(int w, int h)

{

u\_char \*\*img;

int i, j;

if ((img = malloc(sizeof(u\_char\*)\*h)) == NULL){

return NULL;

}

**/\*Function to clear memory of whole image\*/**

for (i = 0; i < h; i++){

img[i] = malloc(w);

if (img[i] == NULL){

for (j = 0; j < i; j++)

free(img[j]);

free(img);

return NULL;

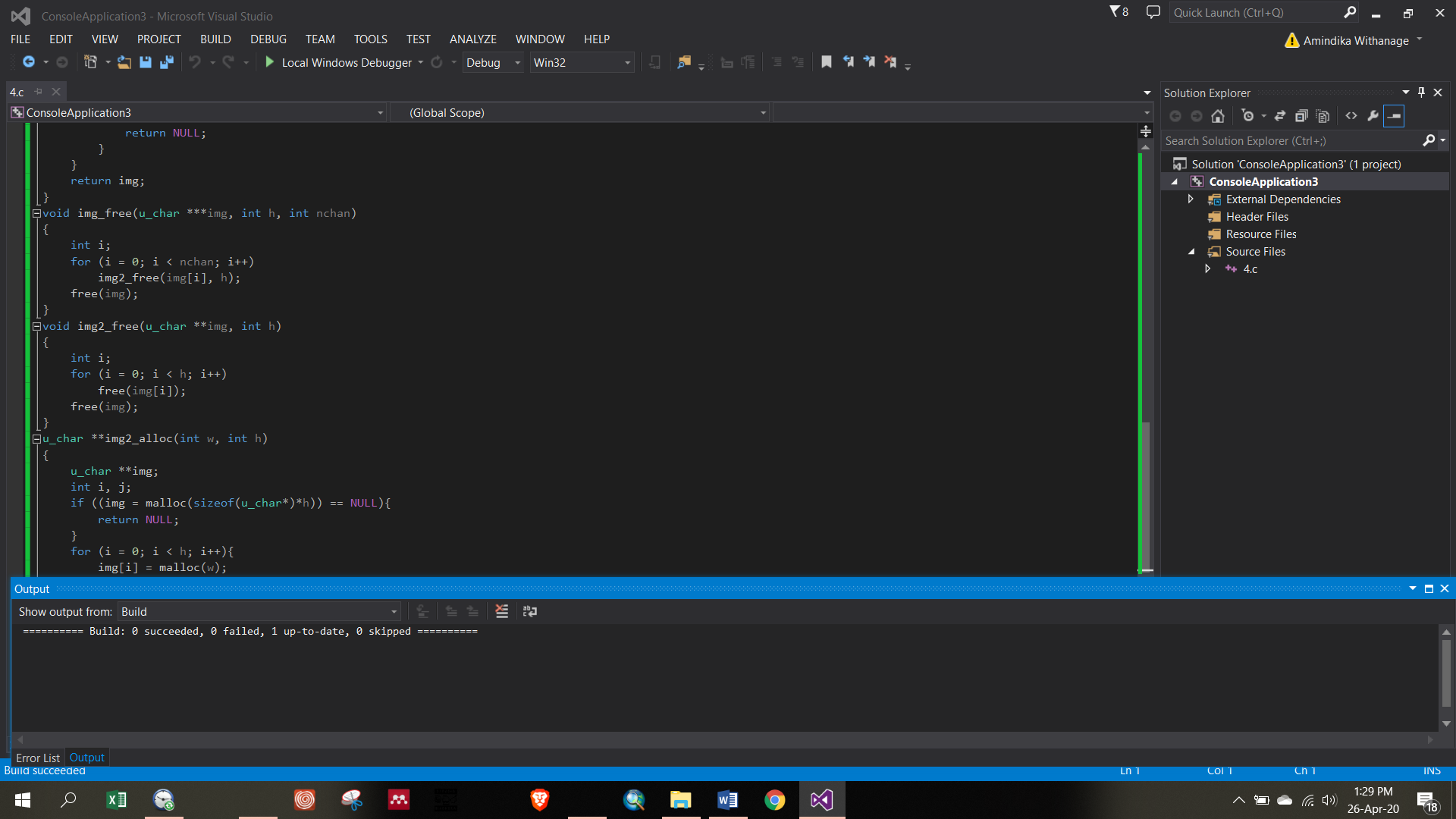
}

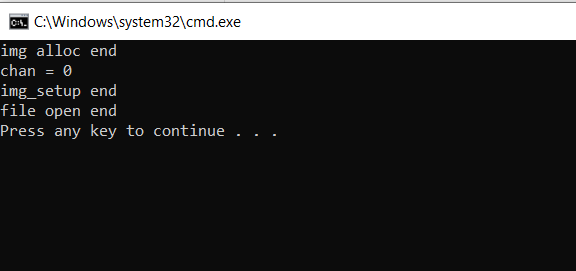
}

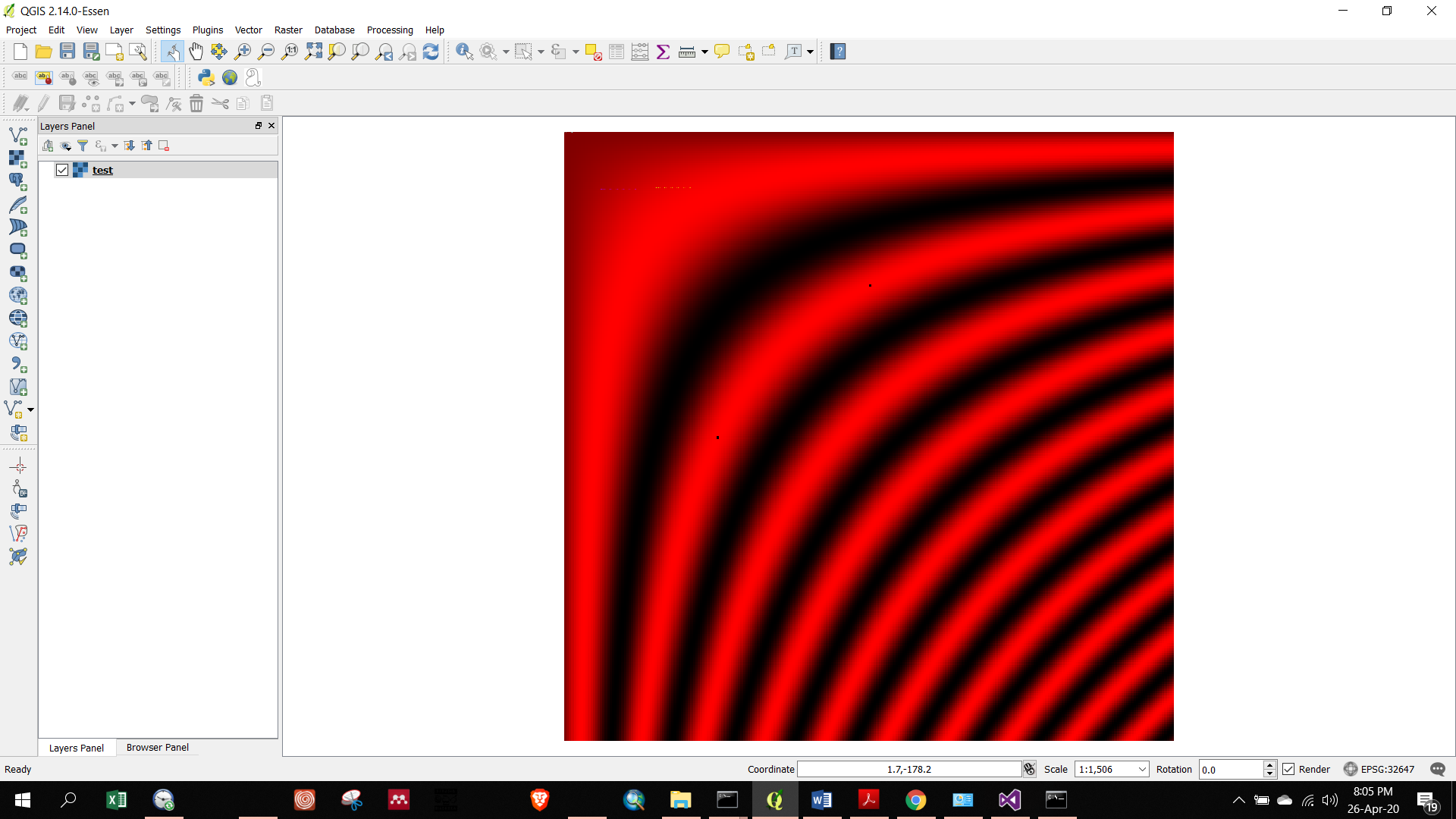
return img;

}

**Answer for A and B).**

Final output is image with W and H= 256, and 3 bands. Output image is a .IMG format which can open from QGIS (***Figure 1*)**



****

**Figure 1:** Color pattern of output test.img

**Answer C).**

**Code description:**

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<malloc.h>

typedef unsigned char u\_char;

typedef unsigned short u\_short;

#define W 256

#define H 256

#define NCHAN 1

#define LUT\_SIZE 256 **//Maximum size of look up table**

void img\_setup(u\_char \*\*\*img, int w, int h, int nchan);

int image\_write(char \*fname, u\_char \*\*\*img, int w, int h, int nchan);

u\_char \*\*\*img\_alloc(int w, int h, int nchan);

void img\_free(u\_char \*\*\*img, int h, int nchan);

u\_char \*\*img2\_alloc(int w, int h);

void img2\_free(u\_char \*\*img, int h);

**//Main function part**

main()

{

u\_char \*\*\*img;

u\_char lut[LUT\_SIZE];

lut\_setup(lut); **//Calling function which are setup a look up table**

img = img\_alloc(W, H, NCHAN); **//Calling allocate memory function**

if (img == NULL){

printf("Not enough memory\n"); exit(1);

}

img\_setup(img, W, H, NCHAN); **//Calling set up function for input “img.img”**

image\_conv(img, W, H, NCHAN, lut); **//Calling conversion function**

image\_write("test.img", img, W, H, NCHAN); **//Calling write function for output “test.img”**

img\_free(img, H, NCHAN); //Calling clear memory function

}

**/\*Function to setup look up table, we just setup only one time for all process\*/**

void lut\_setup(u\_char \*lut)

{

int i;

for (i = 0; i < LUT\_SIZE; i++) {

lut[i] = LUT\_SIZE - i - 1;

}

for (i = 0; i < LUT\_SIZE; i++)

printf("%d\t%d\n", i, lut[i]);

}

**/\*Function for convert pixel value to value in the loof up table\*/**

void image\_conv(u\_char \*\*\*img, int w, int h, int nchan, u\_char \*lut)

{

int i, j, ichan;

for (ichan = 0; ichan < nchan; ichan++)

for (i = 0; i < h; i++)

for (j = 0; j < w; j++)

img[ichan][i][j] = lut[img[ichan][i][j]];

}

**/\*Function to read the input image. Only read image file to program\*/**

int image\_read(char \*fname, u\_char \*\*\*img, int w, int h, int nchan)

{

FILE \*fp;

int i, ichan;

if ((fp = fopen(fname, "rb")) == NULL) {

fprintf(stderr, "image\_read:file open error");

return 1;

}

printf("image\_read:file open end\n");

for (ichan = 0; ichan < nchan; ichan++)

for (i = 0; i < h; i++) {

fread(img[ichan][i], w, 1, fp);

}

fclose(fp);

\_getch( );

return 0;

}

**/\*This part is same as previous code for setting up new image, writing image file**

**output, allocating memory for image and clearing the memory after finish process\*/**

#define PI 3.14159265

void img\_setup(u\_char \*\*\*img, int w, int h, int nchan)

{

int i, j, ichan;

for (ichan = 0; ichan < nchan; ichan++){

printf("chan = %d\n", ichan);

for (i = 0; i < h; i++) {

for (j = 0; j < w; j++) {

**/\* The pattern of image depend on this line od code\*/**  img[ichan][i][j] = sin((double)j / (w\*(1 + ichan\*0.5))\*2.\*PI\*(10.0\*i / h)) \* 128 + 128;

}

}

}

printf("img\_setup end\n");

}

**/\*Function to write the output image\*/**

int image\_write(char \*fname, u\_char \*\*\*img, int w, int h, int nchan)

{

FILE \*fp;

int i, ichan;

**//open file name that we assign in main function**

if ((fp = fopen(fname, "wb")) == NULL){

fprintf(stderr, "image\_write:file open error");

return 1;

}

printf("file open end\n");

**//Write image by each pixel**

for (ichan = 0; ichan<nchan; ichan++)

for (i = 0; i < h; i++)

fwrite(img[ichan][i], w, 1, fp);

fclose(fp);

return 0;

}

**/\*Function to allocate memory for whole image depend on width, height and number of channel\*/**

u\_char \*\*\*img\_alloc(int wdth, int hght, int nchan)

{

u\_char \*\*\*img;

int ichan, j;

**//Using malloc to allocate memory dynamically**

img = malloc(sizeof(u\_char \*\*)\*nchan);

if (img == NULL){

return NULL;

}

for (ichan = 0; ichan < nchan; ichan++){

img[ichan] = img2\_alloc(wdth, hght);

if (img[ichan] == NULL){

for (j = 0; j < ichan; j++)

img2\_free(img[j], hght);

free(img);

return NULL;

}

}

printf("img alloc end\n");

return img;

}

void img\_free(u\_char \*\*\*img, int hght, int nchan){

int i;

for (i = 0; i < nchan; i++)

img2\_free(img[i], hght);

free(img);

}

**/\*Function to allocate memory for each pixel depend on width and height\*/**

u\_char \*\*img2\_alloc(int wdth, int hght)

{

int i, j;

u\_char \*\*img;

img = malloc(sizeof(u\_char \*)\*hght);

if (img == NULL){

return NULL;

}

for (i = 0; i < hght; i++){

img[i] = malloc(sizeof(u\_char)\*wdth);

if (img[i] == NULL){

for (j = 0; j < i; j++)

free(img[j]);

free(img);

return NULL;

}

}

return img;

}

**/\*Function to clear memory each pixel location\*/**

void img2\_free(u\_char \*\*img, int hght)

{

int i;

for (i = 0; i < hght; i++)

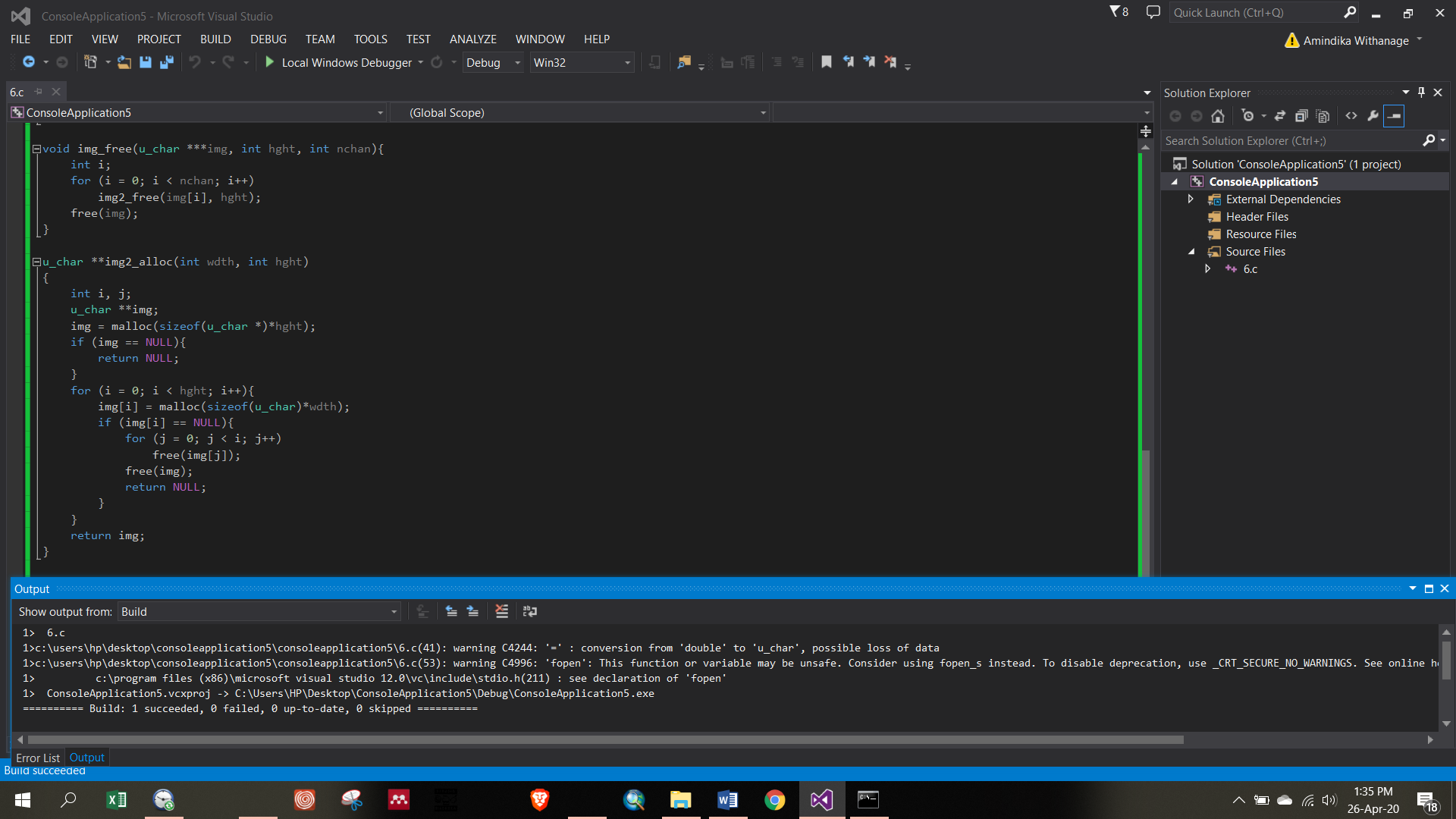
free(img[i]);

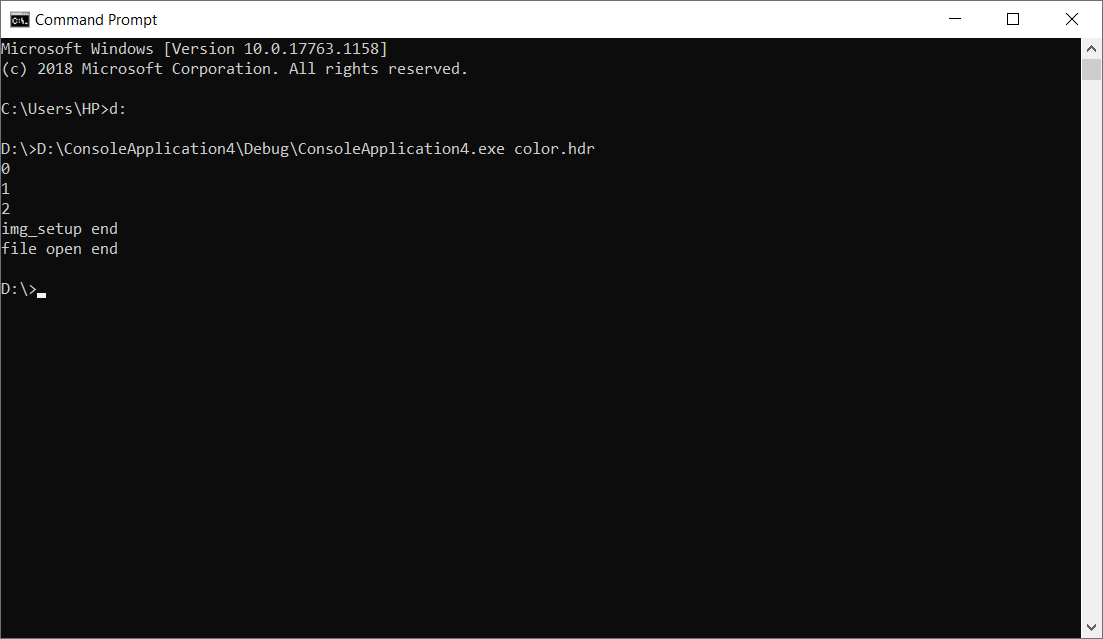
free(img);

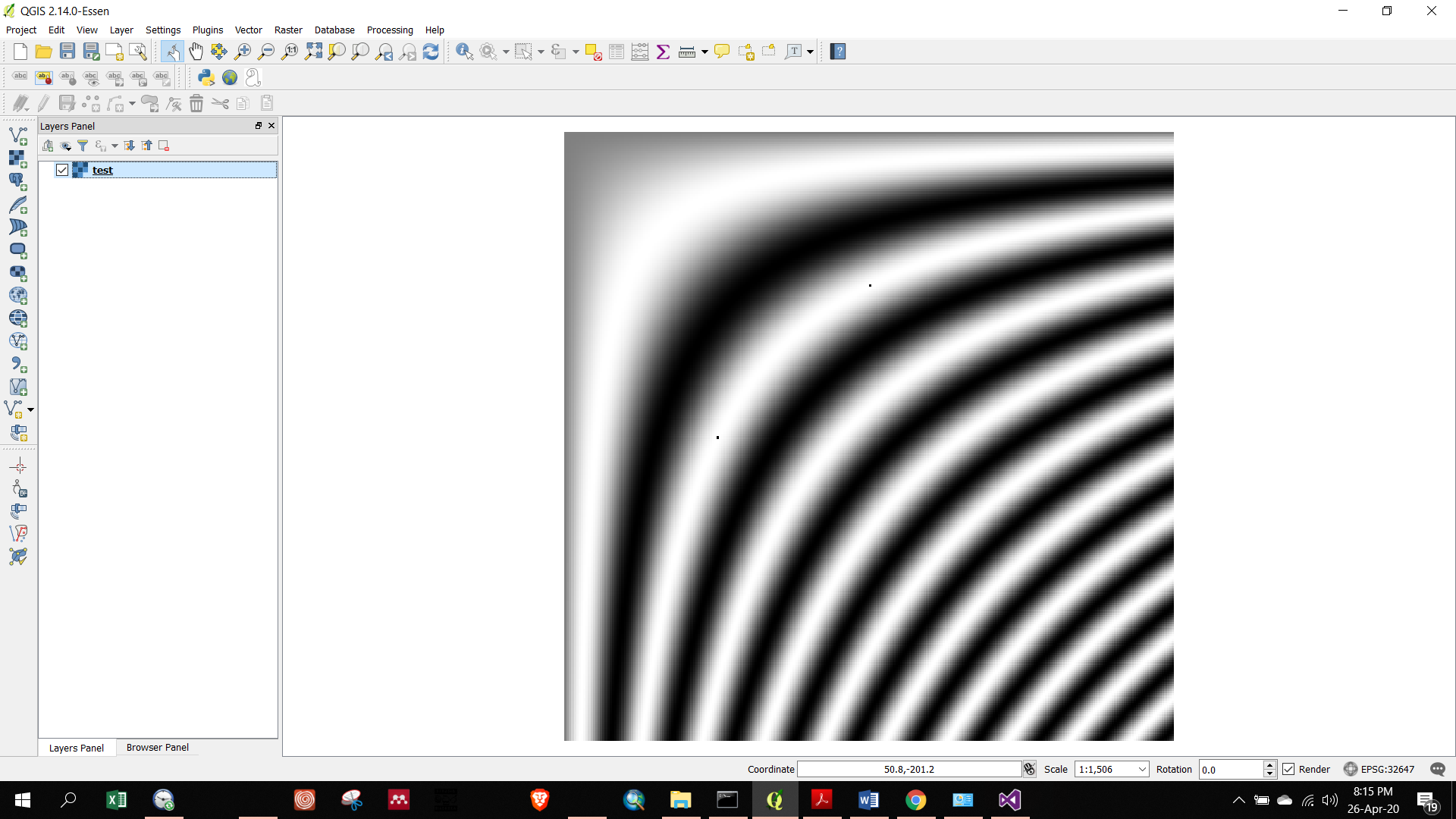
}

**Output:**

Output of new image is .img format which can open in QGIS. It has only one band ***(Figure 2).***







***Figure 2: output 1 band image in gray color pattern (0-255)***