HOMEWORK #2

DUE: April 24, 2020, 17:00 pm

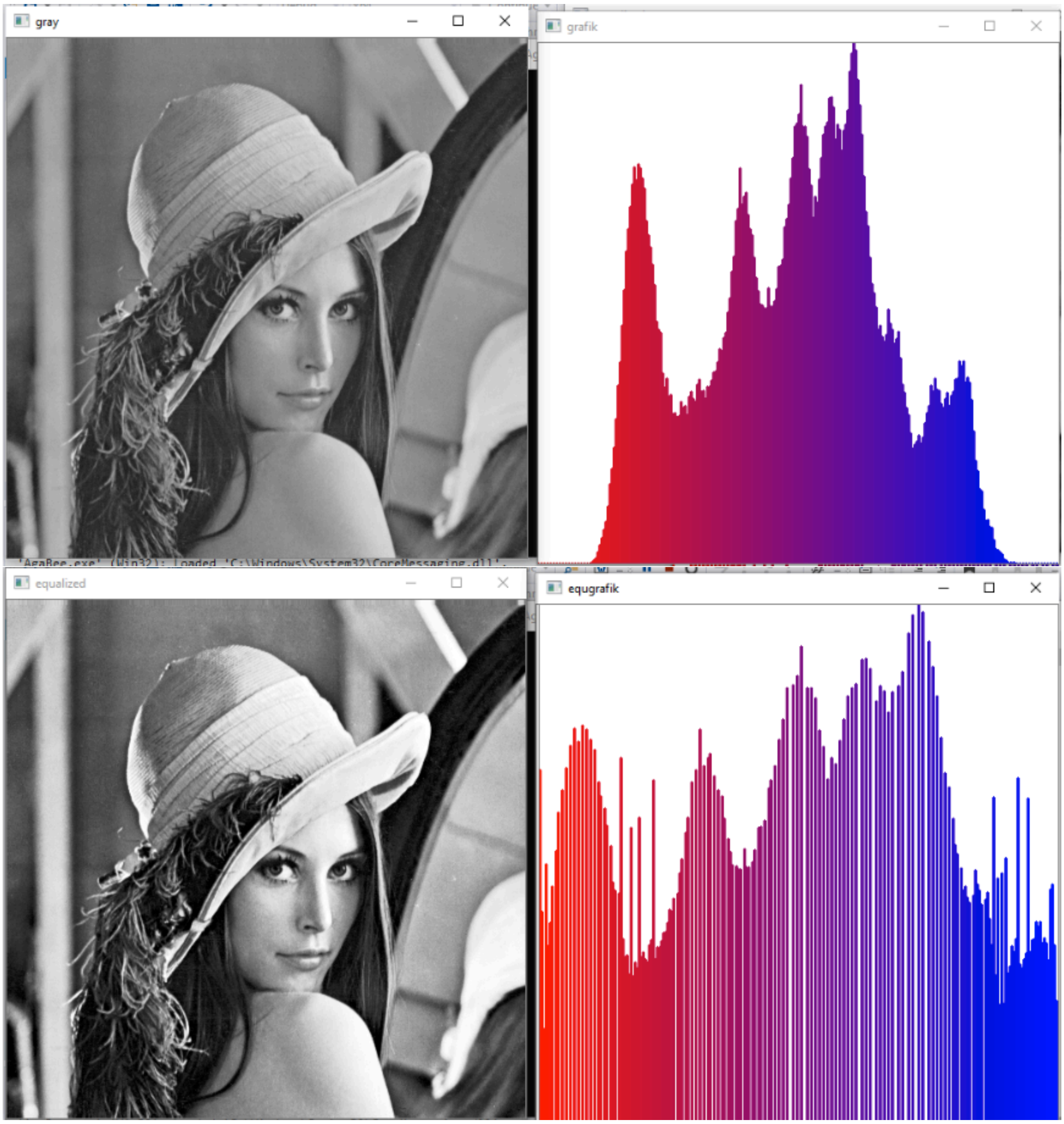
1) Write a raw C code under Visual Studio (with OpenCV added to it) to calculate the histogram of a given grayscale image. Display your histogram if possible.

Code:

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>

using namespace cv;

Mat img;
Mat img2;
Mat img3;
int dizi[255];
int dizi2[255];

int main()
{

       img = imread("lena.png");
       cvtColor(img, img2, CV_BGR2GRAY);
       equalizeHist(img2, img3);
       Mat graph2(512, 512, CV_8UC3, Scalar(255, 255, 255));
       for (int i = 0; i < img3.cols; i++) {
              for (int j = 0; j < img3.rows; j++) {
                     dizi2[img3.at<uchar>(j, i)]++;
              }
       }
       int max2 = dizi2[0];
       for (int i = 1; i < 256; i++) {
              if (dizi2[i] > max2)max2 = dizi2[i];
       }
       for (int i = 1; i < 256; i++) {
              dizi2[i] = dizi2[i] * 512 / max2;
       }
       for (int i = 0; i < 256; i++) {
              line(graph2, Point(i * 2, 512), Point(i * 2, 512 -
dizi2[i]),Scalar(i,0,255-i) ,2);
       }
       ///////////////////7
       Mat graph(512, 512, CV_8UC3, Scalar(255, 255, 255));
       for (int i = 0; i < img2.cols; i++) {
              for (int j = 0; j < img2.rows; j++) {
                     dizi[img2.at<uchar>(j, i)]++;
              }
       }
       int max = dizi[0];
       for (int i = 1; i < 256; i++) {
              if (dizi[i] > max)max = dizi[i];
       }
```

```
        for (int i = 1; i < 256; i++) {
                dizi[i] = dizi[i] * 512 / max;
        }
        for (int i = 0; i < 256; i++) {
                line(graph, Point(i*2 , 512), Point(i*2 , 512 - dizi[i]), Scalar(i, 0, 
255 - i), 2);
        }
        imshow("gray", img2);
        imshow("equalized", img3);
        imshow("grafik", graph);
        imshow("equgrafik", graph2);
        waitKey(0);
        return 0;

}
```

Result:



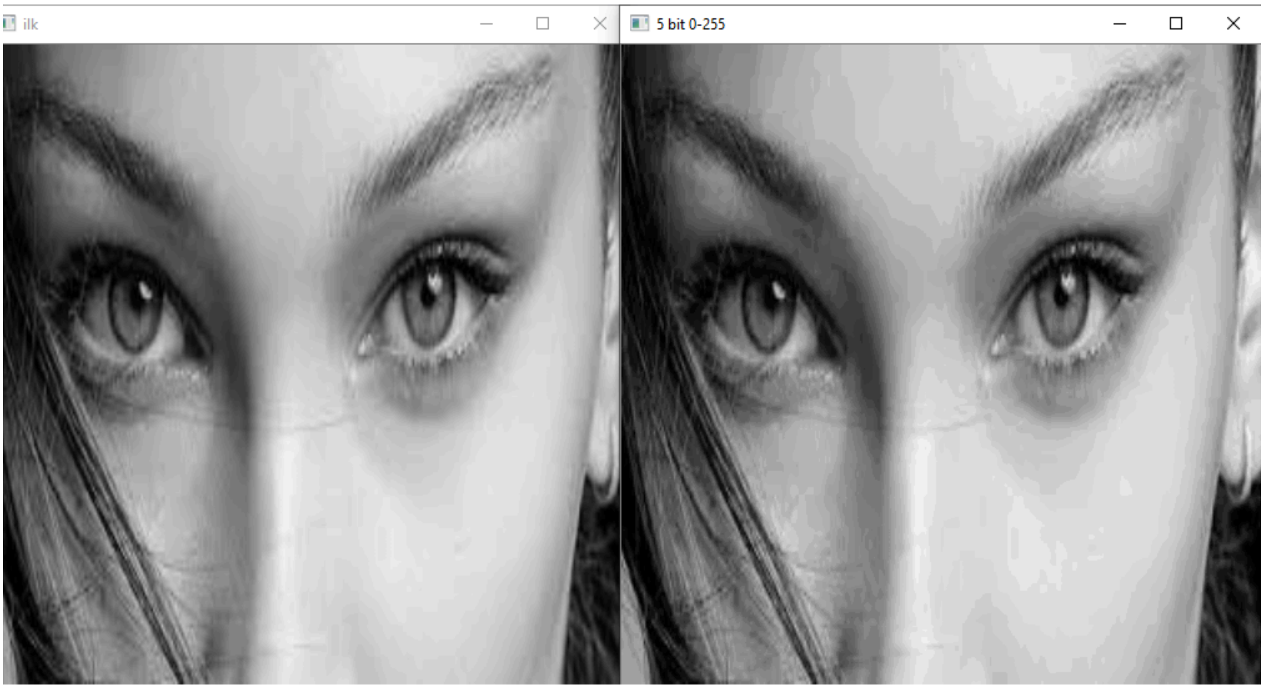
```
'AgaBee.exe' (Win32): Loaded 'C:\Windows\System32\CoreMessaging.dll'.
```

2) Write a raw C code under Visual Studio (with OpenCV added to it) to quantize a given 8-bit grayscae image to 5 bits. Display your result.

Code:

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>
using namespace cv;
Mat img;
Mat img2;
int a;
int main()
{

    img = imread("eyes.jpg");
    cvtColor(img, img2, CV_BGR2GRAY);
    resize(img2, img2, Size(512, 512));
    Mat Grayresim(img2.rows, img2.cols, CV_8UC1, Scalar(0));
    imshow("ilk", img2);

    for (int i = 0; i < img2.cols; i++) {
        for (int j = 0; j < img2.rows; j++) {
            a = 0;
            if (img2.at<uchar>(j, i) % 8 > 4) {
                a = (img2.at<uchar>(j, i) / 8 + 1);
                a = a * 8;
                Grayresim.at<uchar>(j, i) = a;
            }
            else {
                a = img2.at<uchar>(j, i) / 8;
                a = a * 8;
                Grayresim.at<uchar>(j, i) = a;

            }
        }
    }
    imshow("5 bit 0-255", Grayresim);
    waitKey(0);
    return 0;
}
```

Result:



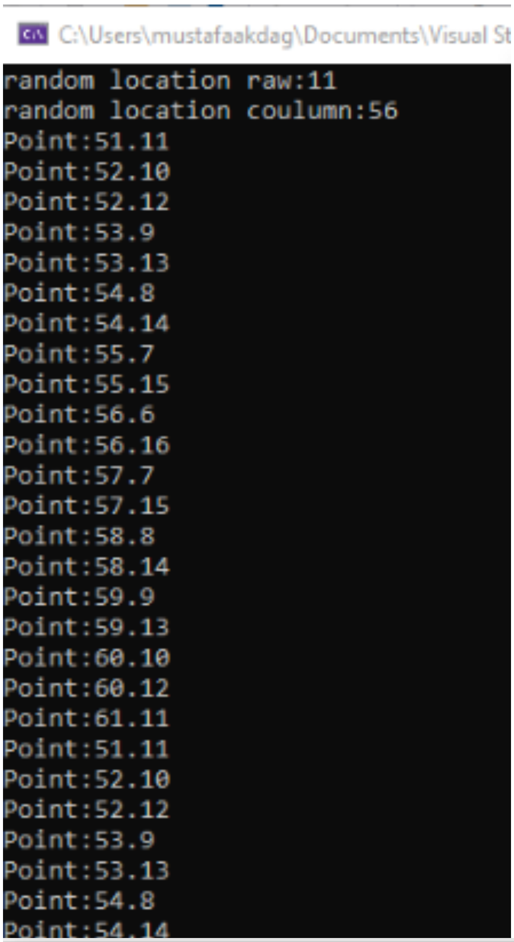
3) Form a 128x128 grayscale image composed of all 0 pixels.

a) Pick a random pixel location within the image.

b) Label all neighboring locations 5 pixels apart the selected pixel as logic level 1. Use the L1 distance for this purpose.

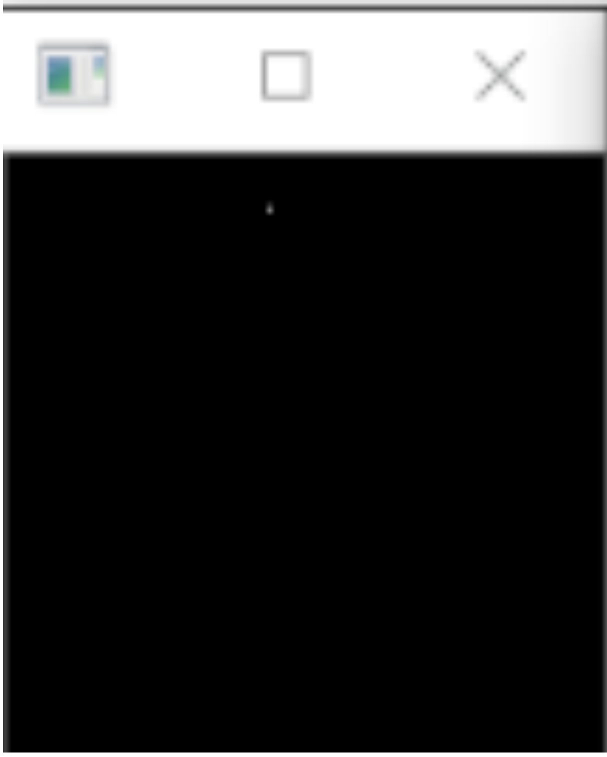
c) Provide the obtained result as an image.

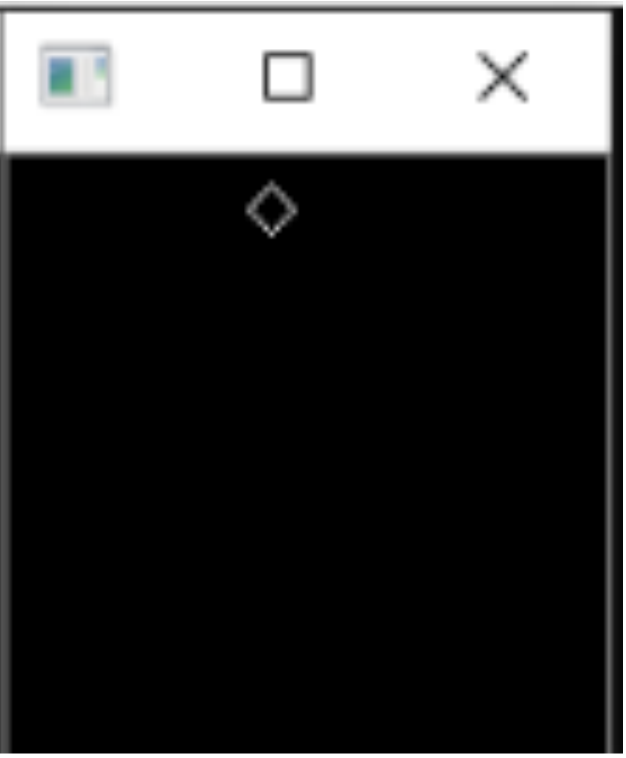
```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
using namespace cv;
using namespace std;
int L = 5;
Point pt1,pt2;
int main()
{
       srand(time(NULL));

       Mat Grayresim(128, 128, CV_8UC1, Scalar(0));
       Mat Grayresim2(128, 128, CV_8UC1, Scalar(0));
       namedWindow("empty image", WINDOW_AUTOSIZE);
       imshow("empty image", Grayresim);
///////part a
       pt1.x = rand() % 127 ;
       pt1.y = rand() % 127;
       cout << "random location raw:" << pt1.y << endl;
       cout << "random location coulumn:" << pt1.x << endl;
       Grayresim.at<uchar>(pt1) = 255;
       namedWindow("ConsideredLocation", WINDOW_AUTOSIZE);
       imshow("ConsideredLocation", Grayresim);
/////////partb
       for (int i = 0; i <= L; i++) {

              for (int j = (pt1.x - L); j <= (pt1.x + L); j++) {

                     for (int k =( pt1.y - L); k <= (pt1.y + L); k++) {

                            if (pt1.x + L < Grayresim2.cols && pt1.x - L > 0 && pt1.y
+ L < Grayresim2.rows && pt1.y - L > 0) {
                                   pt2.x = j;
                                   pt2.y = k;
                                   if ((abs(pt1.x - pt2.x)+ abs(pt1.y - pt2.y))==L) {
                                          Grayresim2.at<uchar>(pt2) = 255;
                                          cout << "Point:" << j << "."<< k <<endl;

                                   }
                            }
                     }
              }

       }
       namedWindow("detectLocation", WINDOW_AUTOSIZE);
       imshow("detectLocation", Grayresim2);

       waitKey(0);
       return 0;

}
////////////part c
```



C:\Users\mustafaakdag\Documents\Visual St

```
random location raw:11
random location coulumn:56
Point:51.11
Point:52.10
Point:52.12
Point:53.9
Point:53.13
Point:54.8
Point:54.14
Point:55.7
Point:55.15
Point:56.6
Point:56.16
Point:57.7
Point:57.15
Point:58.8
Point:58.14
Point:59.9
Point:59.13
Point:60.10
Point:60.12
Point:61.11
Point:51.11
Point:52.10
Point:52.12
Point:53.9
Point:53.13
Point:54.8
Point:54.14
```





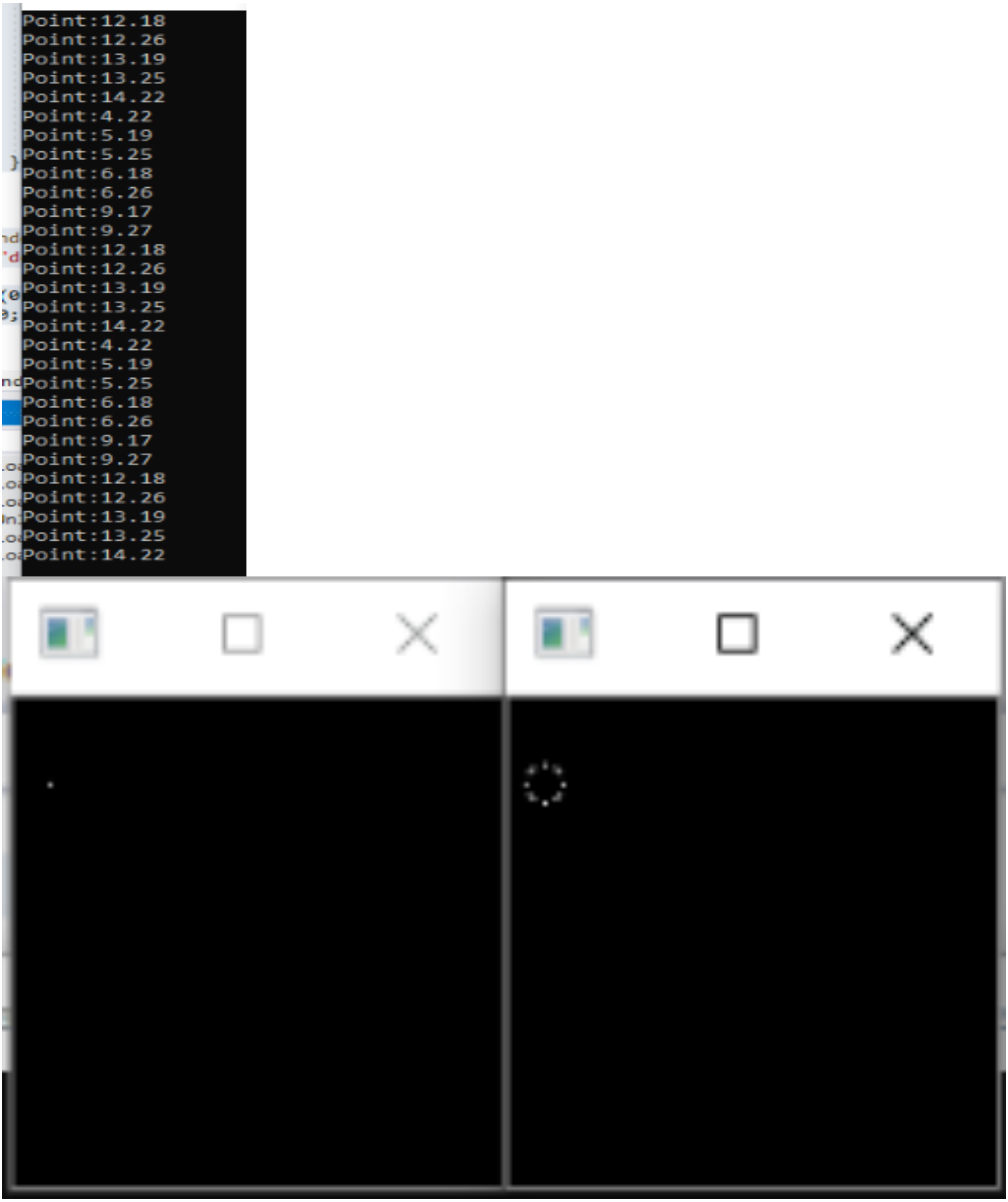
Selected location

L1 distance

4) Repeat Question 3 by using the L2 distance.

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui_c.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
using namespace cv;
using namespace std;
int L = 5;
Point pt1,pt2;
int main()
{
       srand(time(NULL));

       Mat Grayresim(128, 128, CV_8UC1, Scalar(0));
       Mat Grayresim2(128, 128, CV_8UC1, Scalar(0));
       namedWindow("empty image", WINDOW_AUTOSIZE);
       imshow("empty image", Grayresim);

       pt1.x = rand() % 127 ;
       pt1.y = rand() % 127;
       cout << "random location raw:" << pt1.y << endl;
       cout << "random location coulumn:" << pt1.x << endl;
       Grayresim.at<uchar>(pt1) = 255;
       namedWindow("ConsideredLocation", WINDOW_AUTOSIZE);
       imshow("ConsideredLocation", Grayresim);

       for (int i = 0; i <= L; i++) {

              for (int j = (pt1.x - L); j <= (pt1.x + L); j++) {

                     for (int k =( pt1.y - L); k <= (pt1.y + L); k++) {

                            if (pt1.x + L < Grayresim2.cols && pt1.x - L > 0 && pt1.y
+ L < Grayresim2.rows && pt1.y - L > 0) {
                                   pt2.x = j;
                                   pt2.y = k;
                                   if (sqrt((pt1.x - pt2.x)* (pt1.x - pt2.x) + (pt1.y
- pt2.y)* (pt1.y - pt2.y))==L) {
                                          Grayresim2.at<uchar>(pt2) = 255;
                                          cout << "Point:" << j << "."<< k <<endl;

                                   }
                            }
                     }
              }

       }
       namedWindow("detectLocation", WINDOW_AUTOSIZE);
       imshow("detectLocation", Grayresim2);
```

```
        waitKey(0);
        return 0;

}
```



```
Point:12.18
Point:12.26
Point:13.19
Point:13.25
Point:14.22
Point:4.22
Point:5.19
Point:5.25
Point:6.18
Point:6.26
Point:9.17
Point:9.27
Point:12.18
Point:12.26
Point:13.19
Point:13.25
Point:14.22
Point:4.22
Point:5.19
Point:5.25
Point:6.18
Point:6.26
Point:9.17
Point:9.27
Point:12.18
Point:12.26
Point:13.19
Point:13.25
Point:14.22
```