Gebze Technical University
Computer Engineering

CSE 222 - 2018 Spring

HOMEWORK 5 REPORT

ALI HAYDAR KURBAN
151044058

Course Assistant: Ozgu Goksu

1 INTRODUCTION

1.1 Problem Definition

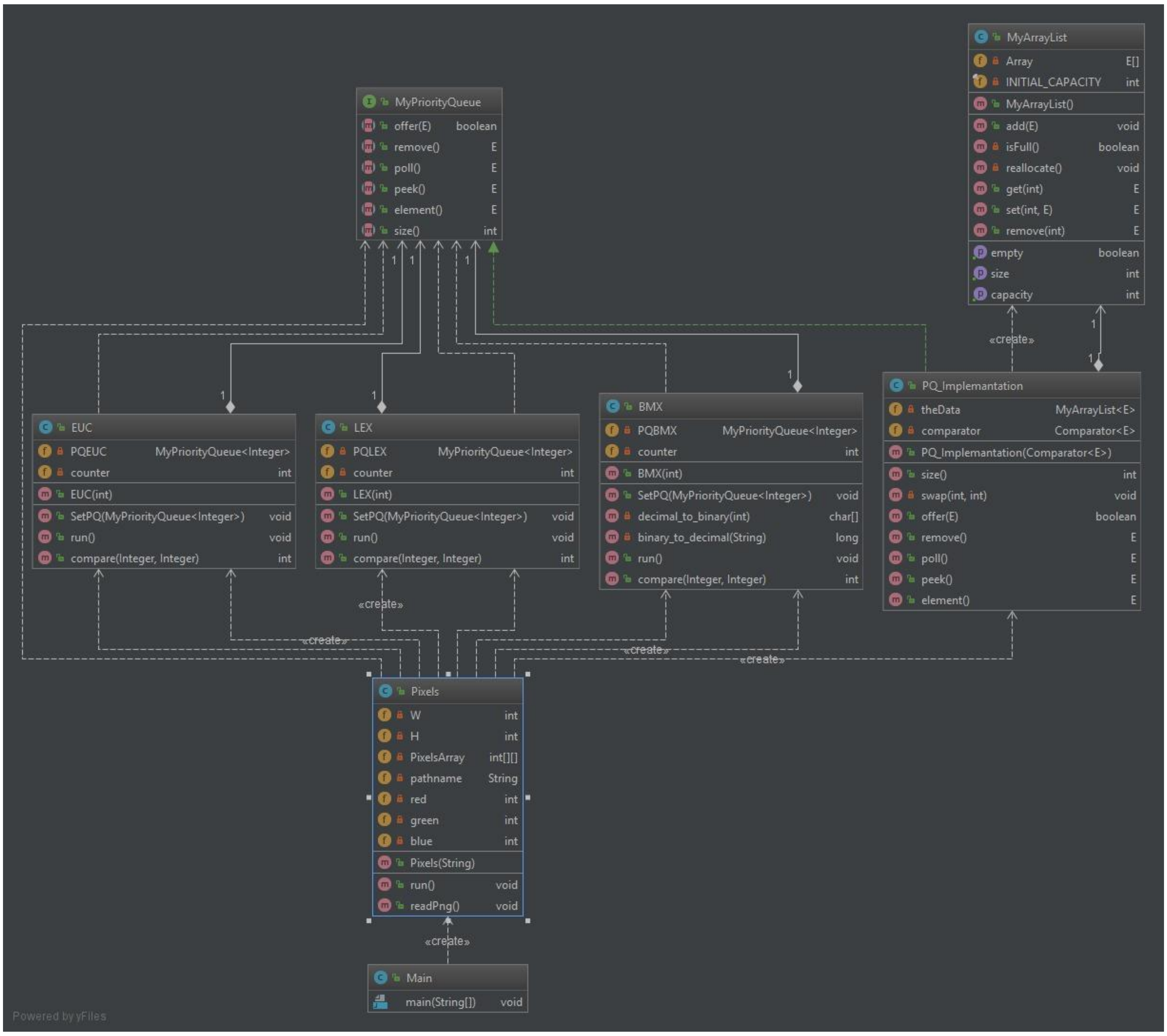
The problem is finding the color of given picture format of png or jpg and add the color in three priority queues. There are three priority which are lexicographical
comparison, euclidean norm based comparison and bitmix comparison. And also we try to solve thread problem, synchronized problem (Parallel Working). Fisrt of all I understood the implementation of priority queues, to implement priority queues we need to compare method because of that I understood the Comparate interface. Then I understood how to create thread, and start them. Finally I understood that how to synchronized threads which are created

1.2 System Requirements

To work my solution for two parts there is some library. These are : java.util.Comparator,
java.util.NoSuchElementException, java.awt.image.BufferedImage, javax.imageio.ImageIO and java.io.FileReader. The solution does not require a speciel piece of hardware. Only it need JVM. The machicne whic has JVM can run this program.

2 METHOD

2.1 Class Diagrams



2.2 Use Case Diagrams

To run this program users have to give a format of png or jpg picture with command line argument and create a thread for Pixels object. The pixcel class's contructor takes an file name from comman line argument and it is your png or jpg picture. It runs like that

```
Thread thread1 = new Thread(new Pixels(args[0]));
        thread1.start();
```

Then program creates priority queues for their priority. And adds pixels and poll them with a synchronize.

2.3 Problem Solution Approach

First of all there is an interface which is shows the methods of Priority Queue. The methods are offer which is add an item to PQ, remove which is delete an item from PQ, poll which is delete an item from PQ, peek is whic is get an item from PQ, element is which is get an item from PQ, size which is return the size of PQ. To implement PQ, I implement my own Array List class it has add, get, set, remove method. These methods are used in PQ implementation. There is a Pixels class it implemets Runnable so that I used this class like Thread1. In this class I read the image and I create 3 PQ with their priority. When I read the image I added all pixel into a 2D integer array. In Pixels class (run method) with a synchronize I offer all pixels to PQ when 100 pixel added other 3 thread started. There is a runnable LEX class, it has compare method. It defines a comparision for PQLEX. And also it has run method, it polls the pixels from PQLEX. There is a runnable EUC class, it has compare method. It defines a comparison dor PQEUC. And also it has run method it polls the pixel from PQEUC. And finally there is a runnable BMX class. In this class there is decimal_to_binary method it takes an positive integer number and convert this number to binary and adds in a char array and returns it. It has binary_to_decimal method it takes a String and convert to long int. It has also compare method . It defines a comparison dor PQBMX. And also it has run method it polls the pixel from PQBMX.

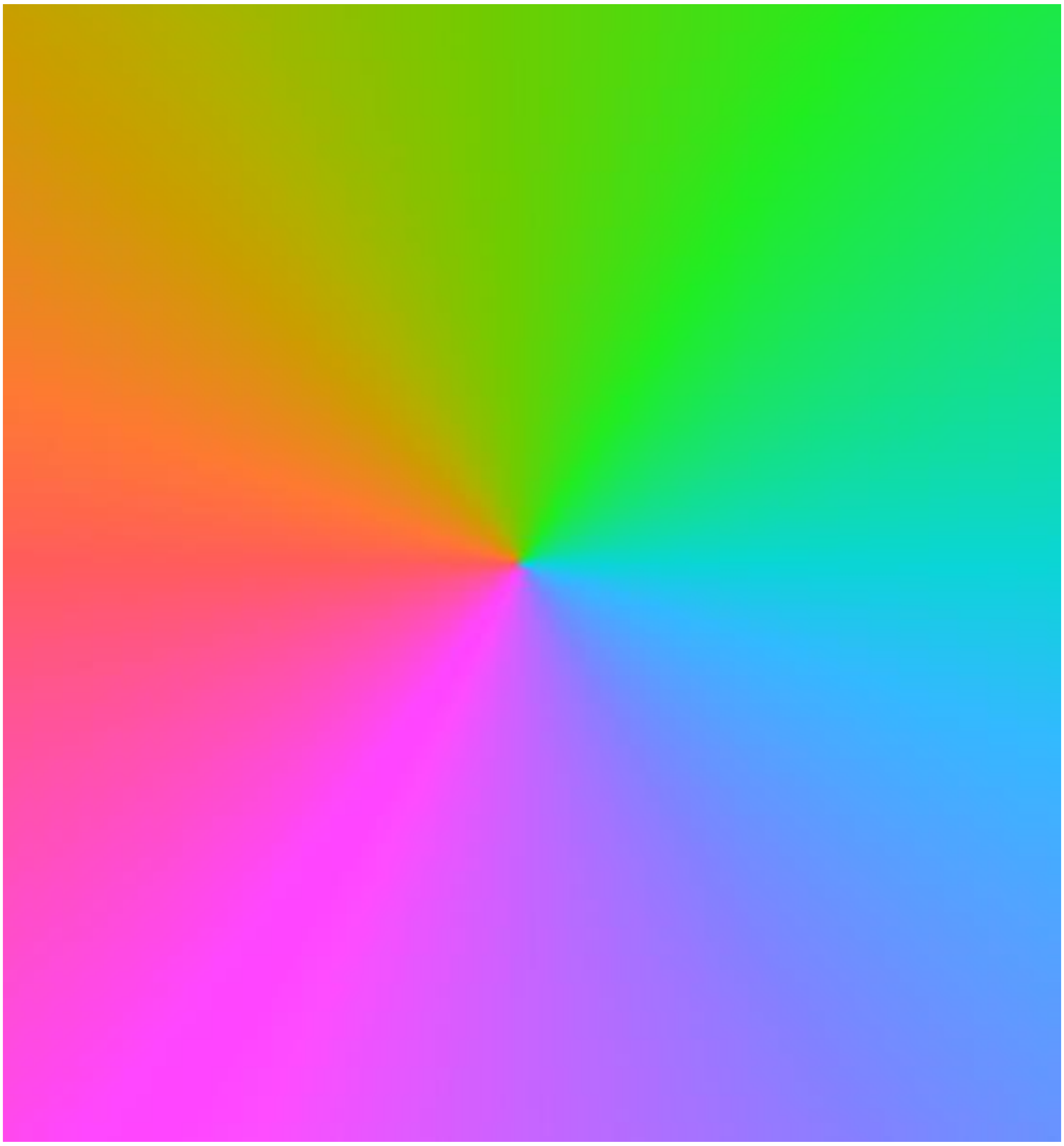
To solve this problem I use Array List which is implemented by me. Because of that it can resize.

The time complexity of adding (offer) PQ is O(log(n)), deleting (poll and remove) element is O(log(n)), geting(peek and element) element is O(1) getting size is O(1).

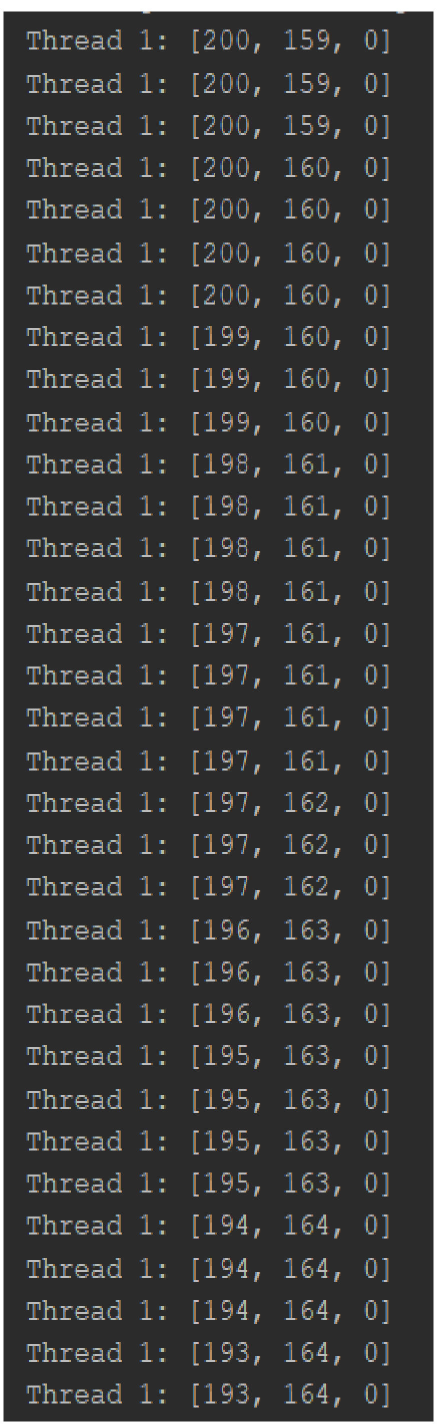
3 RESULT

3.1 Test Cases

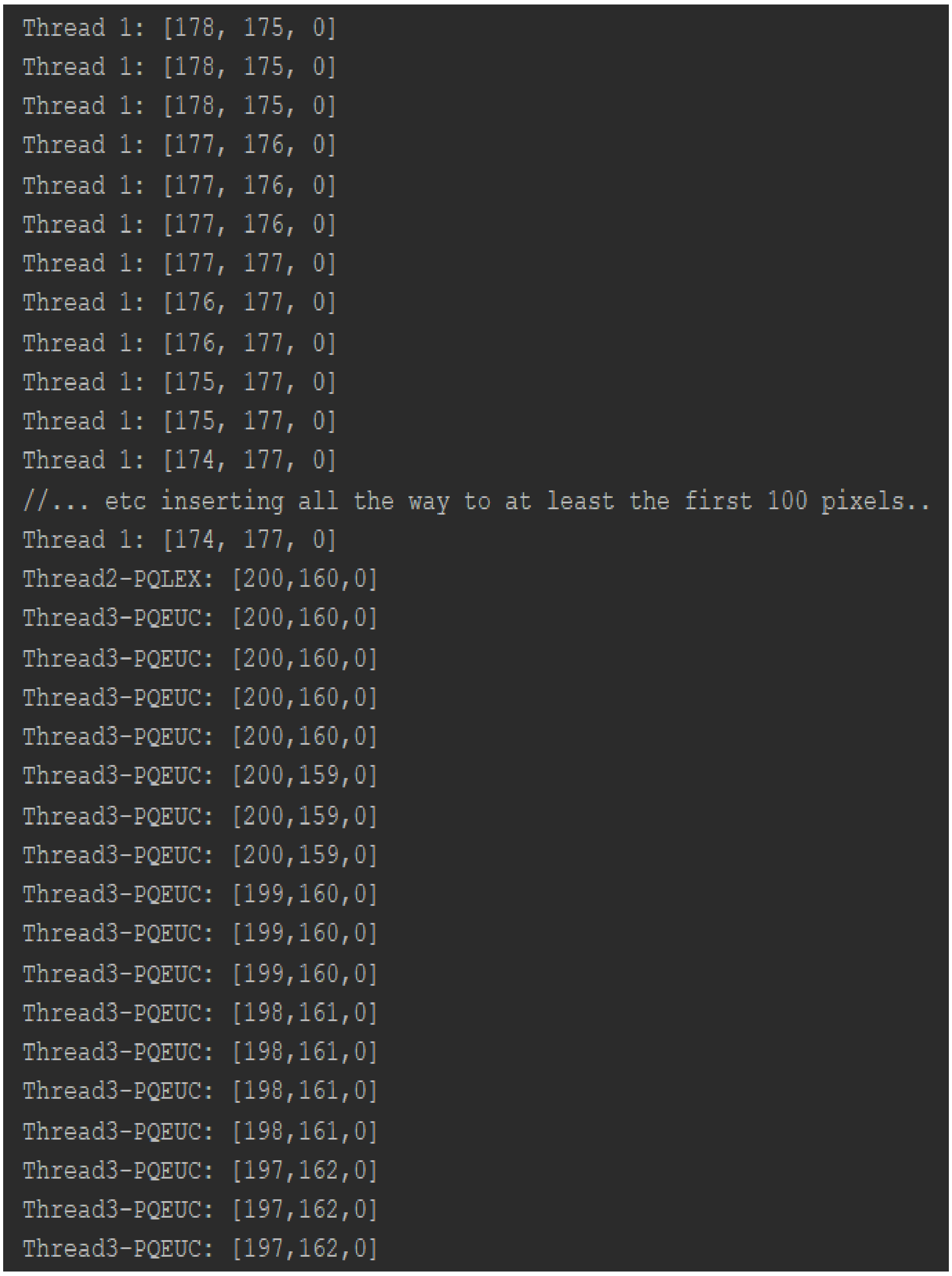
To understant my program is working or not first of all I test all priority queues. First I offered all pixels then I polled all pixels from priority queues. If the first polled pixel is the biggest priority and second pixel is smaller than first polled pixel and like that it means the program worked fine. The test picture is following :



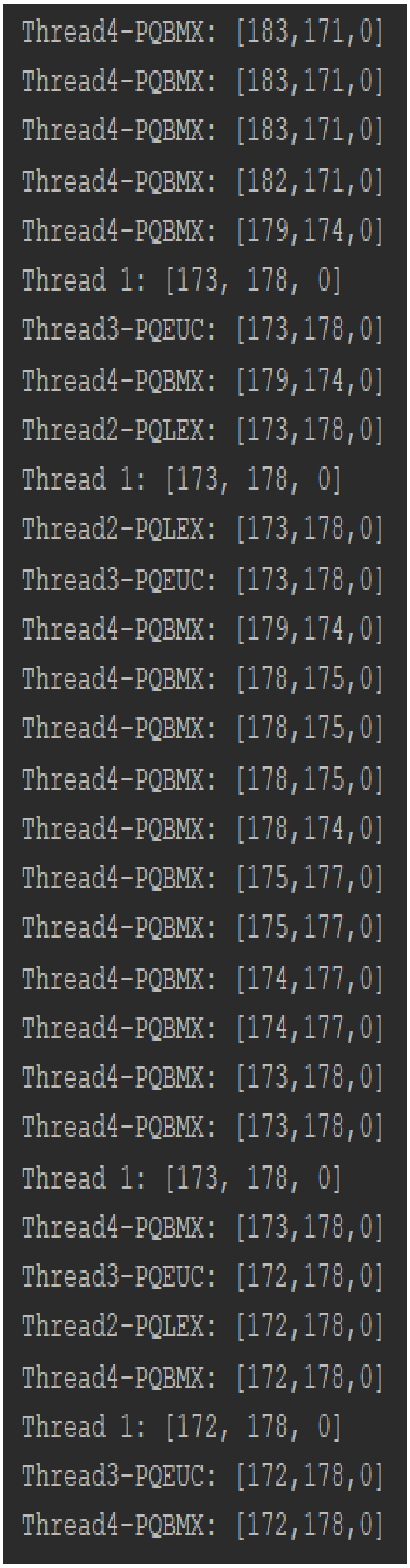
3.2 Running Results



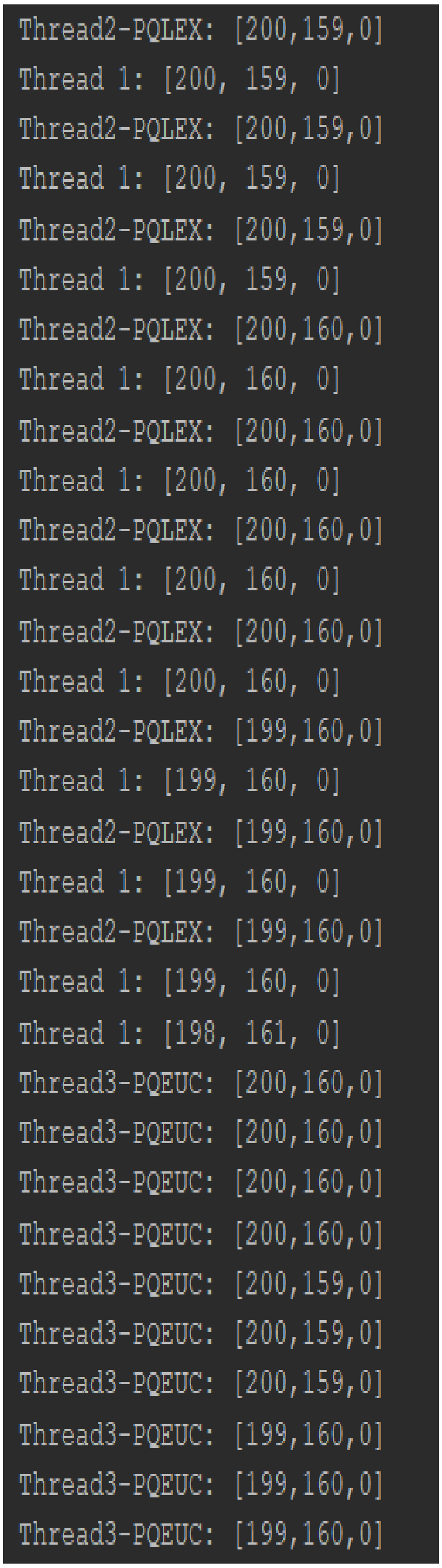
```
Thread 1: [200, 159, 0]
Thread 1: [200, 159, 0]
Thread 1: [200, 159, 0]
Thread 1: [200, 160, 0]
Thread 1: [200, 160, 0]
Thread 1: [200, 160, 0]
Thread 1: [200, 160, 0]
Thread 1: [199, 160, 0]
Thread 1: [199, 160, 0]
Thread 1: [199, 160, 0]
Thread 1: [198, 161, 0]
Thread 1: [198, 161, 0]
Thread 1: [198, 161, 0]
Thread 1: [198, 161, 0]
Thread 1: [197, 161, 0]
Thread 1: [197, 161, 0]
Thread 1: [197, 161, 0]
Thread 1: [197, 161, 0]
Thread 1: [197, 162, 0]
Thread 1: [197, 162, 0]
Thread 1: [197, 162, 0]
Thread 1: [196, 163, 0]
Thread 1: [196, 163, 0]
Thread 1: [196, 163, 0]
Thread 1: [195, 163, 0]
Thread 1: [195, 163, 0]
Thread 1: [195, 163, 0]
Thread 1: [195, 163, 0]
Thread 1: [194, 164, 0]
Thread 1: [194, 164, 0]
Thread 1: [194, 164, 0]
Thread 1: [193, 164, 0]
Thread 1: [193, 164, 0]
```



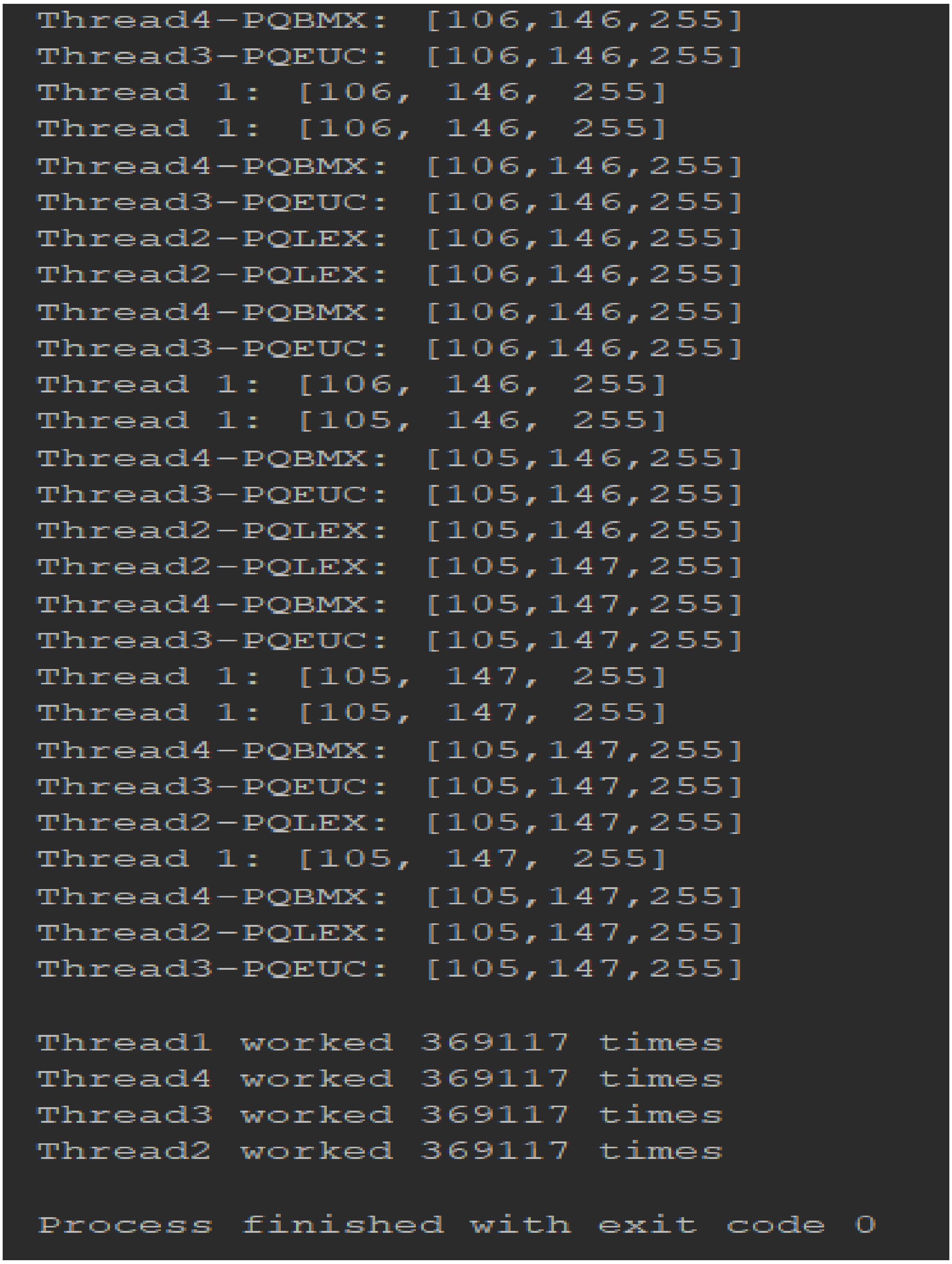
```
Thread 1: [178, 175, 0]
Thread 1: [178, 175, 0]
Thread 1: [178, 175, 0]
Thread 1: [177, 176, 0]
Thread 1: [177, 176, 0]
Thread 1: [177, 176, 0]
Thread 1: [177, 177, 0]
Thread 1: [176, 177, 0]
Thread 1: [176, 177, 0]
Thread 1: [175, 177, 0]
Thread 1: [175, 177, 0]
Thread 1: [174, 177, 0]
//... etc inserting all the way to at least the first 100 pixels..
Thread 1: [174, 177, 0]
Thread2-PQLEX: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,159,0]
Thread3-PQEUC: [200,159,0]
Thread3-PQEUC: [200,159,0]
Thread3-PQEUC: [199,160,0]
Thread3-PQEUC: [199,160,0]
Thread3-PQEUC: [199,160,0]
Thread3-PQEUC: [198,161,0]
Thread3-PQEUC: [198,161,0]
Thread3-PQEUC: [198,161,0]
Thread3-PQEUC: [198,161,0]
Thread3-PQEUC: [197,162,0]
Thread3-PQEUC: [197,162,0]
Thread3-PQEUC: [197,162,0]
```



```
Thread4-PQBMX: [183,171,0]
Thread4-PQBMX: [183,171,0]
Thread4-PQBMX: [183,171,0]
Thread4-PQBMX: [182,171,0]
Thread4-PQBMX: [179,174,0]
Thread 1: [173, 178, 0]
Thread3-PQEUC: [173,178,0]
Thread4-PQBMX: [179,174,0]
Thread2-PQLEX: [173,178,0]
Thread 1: [173, 178, 0]
Thread2-PQLEX: [173,178,0]
Thread3-PQEUC: [173,178,0]
Thread4-PQBMX: [179,174,0]
Thread4-PQBMX: [178,175,0]
Thread4-PQBMX: [178,175,0]
Thread4-PQBMX: [178,175,0]
Thread4-PQBMX: [178,174,0]
Thread4-PQBMX: [175,177,0]
Thread4-PQBMX: [175,177,0]
Thread4-PQBMX: [174,177,0]
Thread4-PQBMX: [174,177,0]
Thread4-PQBMX: [173,178,0]
Thread4-PQBMX: [173,178,0]
Thread 1: [173, 178, 0]
Thread4-PQBMX: [173,178,0]
Thread3-PQEUC: [172,178,0]
Thread2-PQLEX: [172,178,0]
Thread4-PQBMX: [172,178,0]
Thread 1: [172, 178, 0]
Thread3-PQEUC: [172,178,0]
Thread4-PQBMX: [172,178,0]
```



```
Thread2-PQLEX: [200,159,0]
Thread 1: [200, 159, 0]
Thread2-PQLEX: [200,159,0]
Thread 1: [200, 159, 0]
Thread2-PQLEX: [200,159,0]
Thread 1: [200, 159, 0]
Thread2-PQLEX: [200,160,0]
Thread 1: [200, 160, 0]
Thread2-PQLEX: [200,160,0]
Thread 1: [200, 160, 0]
Thread2-PQLEX: [200,160,0]
Thread 1: [200, 160, 0]
Thread2-PQLEX: [200,160,0]
Thread 1: [200, 160, 0]
Thread2-PQLEX: [199,160,0]
Thread 1: [199, 160, 0]
Thread2-PQLEX: [199,160,0]
Thread 1: [199, 160, 0]
Thread2-PQLEX: [199,160,0]
Thread 1: [199, 160, 0]
Thread 1: [198, 161, 0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,160,0]
Thread3-PQEUC: [200,159,0]
Thread3-PQEUC: [200,159,0]
Thread3-PQEUC: [200,159,0]
Thread3-PQEUC: [199,160,0]
Thread3-PQEUC: [199,160,0]
Thread3-PQEUC: [199,160,0]
```



```
Thread4-PQBMX: [106,146,255]
Thread3-PQEUC: [106,146,255]
Thread 1: [106, 146, 255]
Thread 1: [106, 146, 255]
Thread4-PQBMX: [106,146,255]
Thread3-PQEUC: [106,146,255]
Thread2-PQLEX: [106,146,255]
Thread2-PQLEX: [106,146,255]
Thread4-PQBMX: [106,146,255]
Thread3-PQEUC: [106,146,255]
Thread 1: [106, 146, 255]
Thread 1: [105, 146, 255]
Thread4-PQBMX: [105,146,255]
Thread3-PQEUC: [105,146,255]
Thread2-PQLEX: [105,146,255]
Thread2-PQLEX: [105,147,255]
Thread4-PQBMX: [105,147,255]
Thread3-PQEUC: [105,147,255]
Thread 1: [105, 147, 255]
Thread 1: [105, 147, 255]
Thread4-PQBMX: [105,147,255]
Thread3-PQEUC: [105,147,255]
Thread2-PQLEX: [105,147,255]
Thread 1: [105, 147, 255]
Thread4-PQBMX: [105,147,255]
Thread2-PQLEX: [105,147,255]
Thread3-PQEUC: [105,147,255]

Thread1 worked 369117 times
Thread4 worked 369117 times
Thread3 worked 369117 times
Thread2 worked 369117 times

Process finished with exit code 0
```