Jordan University of Science and Technology

Computer Engineering Department

CPE 473: Operating Systems

Programming Assignment : Threading

Notes:

1- **The due date is 26/12/2022 at 11:55 PM.**
2- **Late submissions (i.e. after the due date) will not be accepted.**
3- **Submit the source code files and the PDF report. Write your name and ID at the top**
4- **Groups are allowed (up to 3 students, it is OK to be from different sections)**
5- **Only one submission needed per group. All group members must attend discussion.**

In this assignment, you will implement a multi-threaded program (using C/C++) that will generate a simplified version of the **Image Processing-2D Median Filter** (link). The program will create T worker threads that will operate on the input image and create a new image after applying the filter (T will be passed to the program with the Linux command line). Each of the threads works on a part of the input image. After applying the median filter, each pixel (cell in matrix) will have the value of the median of the original values of all its 8 neighboring pixels and itself (i.e. total of 9 pixels). For pixels in the boundaries where they are missing some neighbors, we can assume the value Zero for these neighbors when calculating the median.

Your program should have three global shared variables to report the statistics of the pixels after applying the filter (i.e. in output image):

1. numOfBright: Will track the total number of pixels that have a value greater than 200.
2. numOfDark: Will track the total number of pixels that have a value less than 50
3. numOfNormal: Will track the total number of pixels that have (x) such that 50<=x<=200

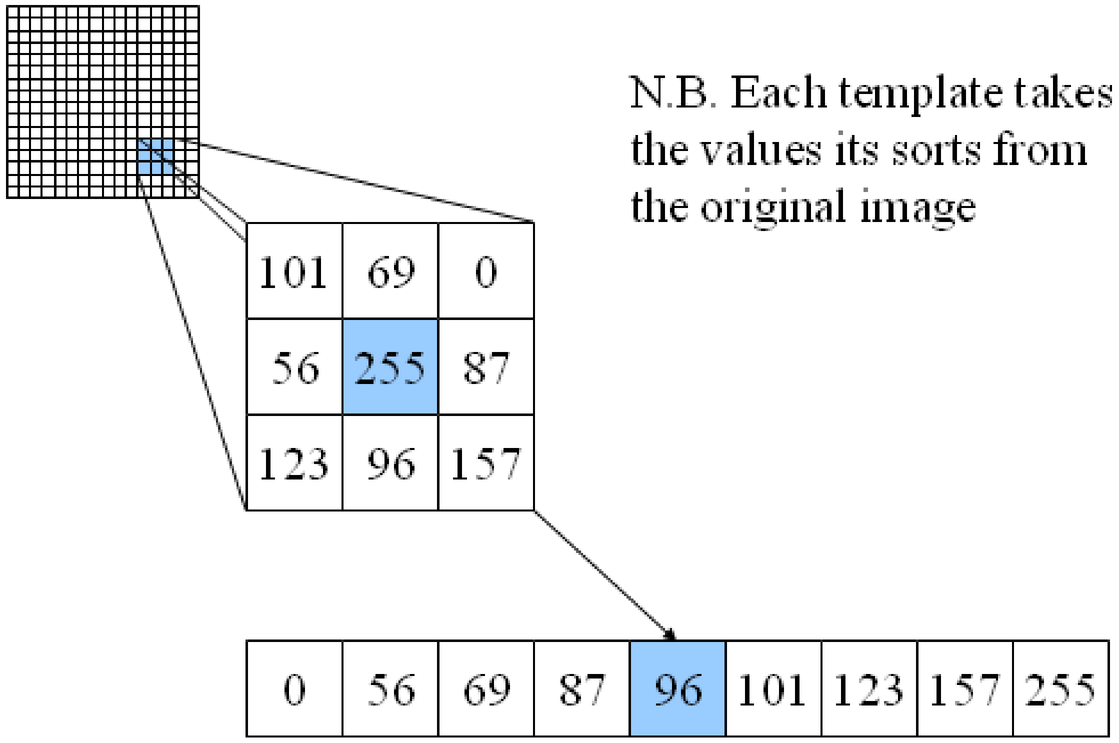
When any of the threads starts executing, it will print its number (0 to T-1), and then the range in the image that it is operating on. Then, when any thread is done, it will print the statistics of pixels it found in its portion. Finally, when all threads are done, the main should print the statistics of all pixels in the entire image.

You should write two versions of the program: The first one doesn't consider race conditions, and the other one is thread-safe.

The input will be provided in an input file (in.txt), and the output should be printed to an output file (out.txt). The number of worker threads (T) will be passed through the command line, as mentioned earlier. The input will start with an integer value N, representing the dimensions of the input image.

Then, an NxN matrix representing the input image (each pixel's value is an integer between 0-255). All the printing messages about statistics (from threads and main) will be printed to the standard output (STDOUT). The resulting output image (matrix) will be printed to the output file (out.txt). See sample example below.

The image below helps visualizing the median filter's operation (source link).


N.B. Each template takes the values its sorts from the original image
101
69
0
56
255
87
123
96
157
0
56
69
87
96
101
123
157
255

Tasks:

In this assignment, **you will submit your source code files for the thread-safe and thread-unsafe versions, in addition to a report (PDF file).** The report should show the following:

1. Screenshot of the main code
2. Screenshot of the thread function(s)
3. Screenshot highlighting the parts of the code that were added to make the code thread-safe, with explanations on the need for them
4. Screenshot of the output of the two versions of your code (thread-safe vs. non-thread-safe), when running passing the following number of threads (T): 1, 4, 16, 64, 256, 1024.
5. Based on your code, how many computing units (e.g. cores, hyper-threads) does your machine have? Provide screenshots of how you arrived at this conclusion, and a screenshot of the actual properties of your machine to validate your conclusion. It is OK if your conclusion doesn't match the actual properties, as long as your conclusion is reasonable.

Hints:

1. **Read this document carefully multiple times to make sure you understand it well. Do you still have more questions? Ask us during our helping sessions, we'll be happy to help!**
2. To learn more about 2D median filter, look at resources over the internet (e.g. link). We only need the parts related to the simple case (window size=9). Think of the image as a 2D matrix.
3. Plan well before coding. Especially on how to divide the range over worker threads. How to synchronize accessing the variables/matrices.
4. For passing the number of threads (T) to the code, you will need to use *argc*, and *argv* as parameters for the main function. For example, the Linux command for running your code with four worker threads (i.e. T=4) will be something like: "./a.out 4"
5. The number of threads (T) and the length of the range can be any number (i.e. not necessarily a power of 2). Your code should try to achieve as much *load balancing* as possible between threads.
6. For answering Task #5 regarding the number of computing units (e.g. cores, hyper-threads) in your machine, search about "diminishing returns". You also might need to use the Linux command "*time*" while answering Task #4, and use input with large matrices (e.g. N>1024).
7. You will, obviously, need to use pthread library and Linux. I suggest you use the threads coding slides to help you with the syntax of the pthread library

Sample Input (in.txt), assuming passing T=4 in the command line:

```
8
215   2    3    240   215   2    3    240
5     100  7    8     5     100  7    8
215   2    3    240   215   2    3    240
215   2    3    240   215   2    3    240
215   2    3    240   215   2    3    240
5     100  7    8     5     100  7    8
215   2    3    240   215   2    3    240
215   2    3    240   215   2    3    240
```

Sample Output (STDOUT terminal part): → Notice that some of threads' output got reordered, that is fine!

```
ThreadID=0, startRow=0, endRow=2
ThreadID=1, startRow=2, endRow=4
ThreadID=1, numOfBright=2, numOfDark=13, numOfNormal=1
ThreadID=2, startRow=4, endRow=6
ThreadID=0, numOfBright=0, numOfDark=15, numOfNormal=1
ThreadID=3, startRow=6, endRow=8
ThreadID=2, numOfBright=0, numOfDark=14, numOfNormal=2
ThreadID=3, numOfBright=0, numOfDark=15, numOfNormal=1
Main: numOfBright=2, numOfDark=57, numOfNormal=5
```

Sample Output (out.txt part):

```
The output image is
0    3    3    5    5    3    3    0
2    5    7    8    100  5    7    3
2    5    7    8    100  5    7    3
2    3    3    215  215  3    3    3
2    5    7    8    100  5    7    3
2    5    7    8    100  5    7    3
2    5    7    8    100  5    7    3
0    2    2    3    2    2    2    0
```