Homework #5: threads (150 pts)
Submit a compressed (.tgz) file with source code and Makefile

• Your job is to implement a multi-threaded (parallel) image-processing program that can transform ANY valid ASCII ppm image in one of seven ways (depending on user input).
  o –I := invert pixels
  o –R := rotate pixels 90° right
  o –L := rotate pixels 90° left
  o –red := keep red pixels only
  o –green := keep green pixels only
  o –blue := keep blue pixels only
  o –C P := adjust contrast by P percent

• The program will be invoked with the following usage:
  UNIX> usage: ./a.out num_threads option [arg]
  o num_threads := positive number of threads
  o option := -I, -R, -L, -red, -green, -blue, or -C
  o arg := decimal between 0 and 1 to specify contrast

• The program will read a ppm file on stdin
• The program will write the converted ppm file to stdout
• Your program must work with any valid ASCII ppm file
• Your program must be written in C (no C++)
• You cannot use the system() function
• PPM images have the following specifications:
  o First line contains “P3”
  o Second line contains two integers: width and height
  o Third line contains one integer: maximum pixel value (maxValue)
  o The remainder of the file contains red, green, blue values for each of the width*height pixels
  o All values are separated by white-space
E.g., 400 x 400 pixel ASCII ppm image of a baboon face
  UNIX> head –n 4 baboon.ppm

P3
400 400
255
177 6 4

• Your program will use threads to do parallel image processing of the ppm file.
  o The image processing will be divided evenly amongst the num_threads threads
    • E.g., if num_threads = N, your program will spawn N threads:
      • thread 1 will process the first (1/N)th of the image
      • thread 2 will process the second (1/N)th of the image
      • ...
      • thread N will process the final (1/N)th of the image
• Examples
  o You may need to install ImageMagick to use the display command
    § UNIX> sudo apt-get install imagemagick
    § (enter password)

UNIX> display baboon.ppm

//use 4 threads to invert pixels; display
UNIX> ./hw4 4 -I < baboon.ppm > output.ppm
UNIX> display output.ppm

//use 4 threads to rotate pixels 90° right; display
UNIX> ./hw4 4 -R < baboon.ppm > output.ppm
UNIX> display output.ppm
//use 4 threads to rotate pixels 90° left; display
UNIX> ./hw4 4 -L < baboon.ppm > output.ppm
UNIX> display output.ppm

//use 4 threads to extract red content only; display
UNIX> ./hw4 4 -red < baboon.ppm > output.ppm
UNIX> display output.ppm

//use 4 threads to extract green pixels only; display
UNIX> ./hw4 4 -green < baboon.ppm > output.ppm
UNIX> display output.ppm
// use 4 threads to extract blue pixels only; display
UNIX> ./hw4 4 -blue < baboon.ppm > output.ppm
UNIX> display output.ppm

// use 4 threads to modify contrast by 80%
UNIX> ./hw4 4 -C 0.80 < baboon.ppm > output.ppm
UNIX> display output.ppm

// You can use pipes to connect multiple image transformations
UNIX> ./hw4 4 -R < baboon.ppm | ./hw4 4 -I | ./hw4 4 -C 0.98 > output.ppm
UNIX> display output.ppm
• Hints:
  o –I (invert) -> \( pixel = maxValue - pixel \)
  o –red -> zero-out green and blue pixels
  o –C P -> if ( pixel <= (maxValue / 2) pixel -= (maxValue*P) else pixel += (maxValue*P)

  o global variables (shared between threads), e.g.,
    • int ***pixels

  o malloc() and free() for a 3D array of pixels (height × width × 3)

  o implement the image transformations first, then refactor to use N threads
    • void* invert(void *arg):

  o Is a mutex really needed?

  o test, test, test
    • Your code must work with ANY valid ppm file (of any size)

  o Draw pictures to help figure out rotation
    • On rotate right, where does pixel 0,0 end up?
    • On rotate left, where does pixel (height-1, width-1) end up?

  o use ImageMagick to convert ANY image file to ASCII ppm

    //convert an image from .jpg to ASCII .ppm
    UNIX> ./convert –compress none image.jpg image.ppm

    NOTE: the conversion may have comments (lines starting with #). Just delete such lines in the ppm file.