Project 3 (C++): You are to implement one of the image enhancements filters: the 5x5 corner preserve averaging as taught in class.

The algorithm can be summarized as follows:

- Mirror frames the image with 4 extra rows and 4 extra columns: 2 rows on the top, 2 rows on the bottom, 2 columns on the left and 2 columns on the right.
- Load the input in framed frameAry.
- Thresholds frameAry using the given threshold value (from argv[2]) and output the threshold result to outFile1
- For every pixel, p, inside the frame, forms 8 regions of pixel p's 5 x 5 neighbors. Say G_1, G_2, …., G_8, where each region includes 9 neighboring pixels of pixel p.
- For each region, G_i, computes the average, A_i, of 9 pixels within the region, resulting 8 averages:
  A_1, A_2, …, A_8, where A_i = 1/9 (sum of all pixels within the region G_i).
  //To simplify the averaging computation, you may use 8 5x5 masks, say, M_1, M_2, …, M_8, where each M_i will have 9 pixels of 1's in the corresponding location of pixels in the region of G_i and all other 16 pixels outside of region will set to 0. (Masks are given in this project.) The convolution operation is performed on each region using the corresponding mask, then divide by 9 to get the average A_i

  A_i = 1/9 (convolute G_i with M_i), i = 1, 2, …, 8

- Determine the absolute differences between p's value and each Ai
- outAry ← Ai where Ai having absolute differences, i.e., is most similar to p's value
- Threshold outAry using the given threshold value
- Pretty print the outAry (with frame) to outFile1 without header.
- output thresholded mage header and all pixels inside of the frame of thrAry to outFile2.

****

Project points: 10 pts
Language: C++
Due Date: Soft copy (*.zip) and hard copies (*.pdf):

+1 (11/10 pts): early submission, 2/26/2022, Saturday before midnight
-0 (10/10 pts): on time, 3/2/2022 Wednesday before midnight
-1 (9/10 pts): 1 day late, 3/3/2022 Thursday before midnight
-2 (8/10 pts): 2 days late, 3/4/2022 Friday before midnight
(-10/10 pts): non submission, 3/4/2022 Friday after midnight

*** Name your soft copy and hard copy files using the naming convention as given in the project submission requirement.
*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in **the same email attachments** with correct email subject as stated in the email requirement; otherwise, your submission will be rejected.

==========

You will be given a grey-scale image data1 and 8 masks.
1. Implement your program as given the specs below.
2. Run your program with the data
3. Include in your hard copy *.pdf file as follows:
   - Cover page.
   - Source code.
   - outFile1
   - outFile2

****

I. Inputs:

a) inFile (argv[1]): A text file representing a grey-scale image with image header.
b) thrVal (argv[2]): Threshold value. Try 30. use atoi (argv[2]) to get integer from argv.

****

II. Outputs: There are two output files

a) outFile1 (argv[3]: all outputs the program dictates
b) outFile2 (argv[4]): The threshold smoothed image.

****
III. Data structure:
****

- An imageProcessing class
    - (int) numRows
    - (int) numCols
    - (int) minVal
    - (int) maxVal
    - (int **) frameAry // a 2D array to store input image, dynamically allocate, size of numRows+4 by numCols+4.
    - (int **) outAry // a 2D array to store the result of the method, dynamically, size of numRows+4 by numCols+4.
    - (int **) thrAry // a 2D array to store the result of thresholding, dynamically, size numRows+4 by numCols+4.
    - (int) thrVal // from argv[]
    - (int) mask[8][5][5] // This 3D array is used in convolution. You may hard code the array according the
        // 8 mask files or read all 8 masks as files.

methods:

- loadImage (...) // load the input image to frameAry, inside framing, begins at (2, 2).
- mirrorFraming (...) // On your own. The algorithm of Mirror framing was taught in class
- loadMask (…) // mask[8][5][5] ←load 8 mask arrays from files or hard code the 8 masks
- (int) convolution5x5 (i, j, mask) // On your own. Perform convolution on frameAry (i,j)'s 5 by 5 neighbors with
    // mask, then returns the convolution result. Use indexes in loops! -5 pts if you write 25 lines of codes.
- cornerPreserveAvg (…) // See algorithm below.
- threshold (outAry, thrAry, thrVal) // if outAry [i, j] >= thrVal
    // thrAry [i, j] ← 1
    // else
    // thrAry [i, j] ← 0
- imgReformat (...) // see algorithm below.

****
IV. main(...)
****

Step 0: inFile, outFile1, outFile2 ← open from agrv[]
numRows, numCols, minVal, maxVal ← read from inFile
thrVal ← get from argv[]
frameAry, outAry, thrAry ← Dynamically allocated

Step 1: loadImage (frameAry, inFile)
mirrorFraming (...)
loadMask (…)

Step 2: imgReformat (frameAry, minVal, maxVal, outFile1)
Step 3: threshold (outAry, thrAry, thrVal)

Step 4: imgReformat (thrAry, 0, 1, outFile1)

Step 5: cornerPreserveAvg (…)

Step 6: imgReformat (outAry, minVal, maxVal, outFile1)

Step 7: threshold (outAry, thrAry, thrVal)

Step 8: imgReformat (thrAry, 0, 1, outFile1)

Step 9: outFile2 ← output image header (numRows, numCols, 0, 1) and all pixels inside frame thrAry to outFile2.

Step 10: close all files.

```
*******************************
V. cornerPreserveAvg (…)
*******************************
Step 1:  r ← 2
         c ← 2
Step 2:  maskIndex ← 0
         minAvg ← frameAry [r, c]
         minDiff ← 9999

Step 3:  result ← convolution5x5 (r, c, mask[maskIndex]) / 9
Step 4:  diff ← abs (result  - frameAry (r, c) )
Step 5: if diff < minDiff
             minDiff ← diff
             minAvg ← result
Step 6:  maskIndex ++
Step 7:  repeat Step 3 to Step 6 while maskIndex < 8

Step 8:  outAry[r, c] ← minAvg
Step 9:  c++
Step 10: repeat Step 2 to Step 9 while c < numCols + 2
Step 11: r++
Step 12: repeat Step 2 to Step 11 while r < numRows + 2

*******************************
VI. imgReformat (inAry, newMin, newMax, OutFile)
*******************************
        // make sure you have #include<string>
Step 1: OutFile ← output numRows, numCols, newMin, newMax
Step 2: str ← to_string(newMax)  // a method in C++ string class
           Width ← length of str
Step 3: r ← 1
Step 4: c ← 1
Step 5: OutFile ← inAry[r][c]
Step 6: str ← to_string (inAry[r][c])
           WW ← length of str
Step 7:  OutFile ← one blank space
           WW ++
Step 8: repeat step 7 while WW < Width
Step 9: c++
Step 10: repeat Step 5 to Step 9 while c <= numCols
Step 11: r++
Step 12: repeat Step 4 to Step 10 while r <= numRows
```