Project 6 (in C++): Given a binary image, the task is to produce a loss-less compression via the skeleton of 8-connectness distance transform. To insure the correctness of your compression, you will do a decompression of the compressed file.

Summary of what your program will do:

1) Load input into inside of the frame of ZFAry (zero-framed).
2) Performs the 1st-pass of the 8-connectness distance transform for all pixels inside the frame of ZFAry.
3) reformatPrettyPrint of the result of the Pass-1 to outFile1 with proper captions.
4) Performs the 2nd-pass of the 8-connectness distance transform on the result of 1st pass (inside of the frame)
5) reformatPrettyPrint of the result of the Pass-2 to outFile1 with proper captions.
6) Performs local maxima operation on the result of 2nd-pass.
7) reformatPrettyPrint the local maxima to outFile1 with proper captions.
8) Produce skeleton (compressed file)
9) The name of the compressed file is to be created during the run time of your program, using the original file name with an extension "_skeleton." For example, if the name of the input file is "image", then the name of the compressed file should be "image_skeleton".
10) close the compressed file.
// To make sure your program works correctly; you are going to do a de-compression on the compressed file as follows.
11) re-open the compressed file (image_skeleton).
12) re-set ZFAry to zero
13) Load triplets from compressed file to ZFAry
14) Perform 1st-pass expansion on the ZFAry
15) reformatPrettyPrint of the result of 1st-pass expansion to outFile1 with captions.
16) Perform 2nd pass expansion on the result of 1st expansion
17) reformatPrettyPrint of the result of 2nd-pass expansion to outFile1 with caption.
    // If your program work correctly, the result of 2nd-pass expansion should be
    // identical to the result of the 2nd pass of distance transform.
18) Produce decompressed file:
    a) Write the original image header to the decompressed file
    b) Threshold ZFAry with threshold value == 1 begins at (1,1)
        and ends at (?,?)
      i.e., if ZFAry (i, j) >= 1
                output 1 and a blank space to de-compressed file.
            else
                output 0 and a blank space to de-compressed file.
19) The name of the decompressed file is to be created during the run time of your program, using the name of the input file with an extension "_decompressed." For example, if the name of the input file is "image", then the name of the compressed file should be "image_decompressed". (This can be done simply using string concatenation.)
20) Closed the de-compressed file.
  // after this step your directory should have these three files: image, image_skeleton, and image_decompressed.
21) If your program works correctly, image_decompressed should be identical to image.

*** You will be given 2 data files: img1 and img2. img1 is a small and simple image. img2 is a larger image.
What to do as follows:
1) Implement your program based on the specs given below.
2) Run and debug your program with img1 until your program produces the decompress file is identical to img1.
3) When the result is correct, then run your program with img2.

Include in your hard copies:
- cover page
- source code
- For img1
    - Print the input file
    - Print outFile1
    - Print skeletonFile
    - Print decompressFile
- For img2
    - Print the input file

- Print outFile1
- Print skeletonFile
- Print decompressFile

****

Language: C++

****

Project points:12 pts

Due Date: Soft copy (*.zip) and hard copies (*.pdf):

-0 (13/12 pts): early submission, 11/30/2022 Sunday before midnight
+1 (12/12 pts): early submission, 11/3/2022, Thursday before midnight
-1 (11/12 pts): 1 day late, 11/4/2022, Friday before midnight
-2 (10/12 pts): 2 days late, 11/5/2022, Saturday before midnight
(-12/12 pts): non submission, 11/5/2022, Saturday 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.

****

I. inFile (argv[1]): a binary image with image header.

****

II. Outputs:

a) outFile1 (argv [2]): for
- reformatPrettyPrint of the results of 1st pass 8-connectness distance transform
- reformatPrettyPrint of the results of 2nd pass 8-connectness distance transform
- reformatPrettyPrint of the local maxima skeleton
- reformatPrettyPrint of the results of 1st pass expansion
- reformatPrettyPrint of the results of 2nd pass expansion

b) skeletonFile (the name of the file is generated at run-time) for store the compressed file using the following format:

Example:

```
20 20 0 9 // the header of the distance transform image.
4 7 2     // the skeleton pixel at (4, 7) with distance of 2
6 7 3     // the skeleton pixel at (6, 7) with distance of 3
:
:
```

c) deCompressFile (the name of the file is generated at run-time), an image file where the first text-line is the image header, follows by rows and cols of pixel values.

****

III. Data structure:

****

- An ImageCompression class
  - (int) numRows
  - (int) numCols
  - (int) minVal
  - (int) maxVal
  - (int) newMinVal
  - (int) newMinVal
  - (int **) ZFAry //a 2D array, need to dynamically allocatevof size numRows + 2 by numCols + 2.
  - (int **) skeletonAry //a 2D array, need to dynamically allocate of size numRows + 2 by numCols + 2.

- methods:
  - setZero (Ary) // set 2D Ary to zero.
  - loadImage (…) // Load input onto inside frame of ZFAry.
  - loadSkeleton (…) // Load the skeleton file onto inside frame of ZFAry.

- Compute8Distance (...) // See algorithm below.
- fistPass8Distance (…) // algorithm is given in class.
- secondPass8Distance (…) // algorithm is given in class.
  // Note** In second pass, you need to keep track the newMinVal and newMaxVal.
- imageCompression (...) // See algorithm below
- isLocalMaxima (…) // A pixel is local maxima if >= to all its 8 neighbors. On your own
- computeLocalMaxima (ZFAry, skeletonAry) // Check all pixels, ZFAry[i,j] in ZFAry
  // if isLocalMaxima(ZFAry[i,j])
  skeletonAry[i,j] ← ZFAry[i,j]
  else skeletonAry[i,j] ← 0
- extractSkeleton (...) // if skeletonAry[i,j] > 0 write the triplet (i.e., i, j, skeletonAry[i,j]) to skeletonFile.
  // Please note, in real life, i and j need to subtract by 1 since skeletonAry has been framed;
  // however, for easy programming and since we are reusing ZFAry,
  // i and j do not need to subtract by 1.
- imageDeCompression (...) // See algorithm below
- firstPassExpension (...)// algorithm is given in class.
- secondPassExpension (...)// algorithm is given in class.
- threshold (...) // do a threshold on pixels inside of ZFAry with the threshold value at 1;
  i.e., if ZFAry (i, j) >= 1
  output 1 and a blank space to decompressed file.
  else
  output 0 and a blank space to decompressed file.
- reformatPrettyPrint (…) // reuse codes from your previous project.

****
VI. main (…)
****

step 0: open inFile and outFile1
numRows, numCols, minVal, maxVal ← read from inFile
dynamically allocate ZFAry with extra 2 rows and 2 cols
dynamically allocate skeletonAry with extra 2 rows and 2 cols

Step 1: skeletonFileName ← argv[1] + "_skeleton.txt"
Step 2: skeletonFile ← open (skeletonFileName)
Step 3: deCompressedFileName ← argv[1] + "_deCompressed.txt"
Step 4: deCompressFile ← open (deCompressedFileName)
Step 5: setZero (ZFAry)
setZero (skeletonAry)
Step 6: loadImage (inFile, ZFAry)
Step 7: compute8Distance (ZFAry, outFile1) // Perform distance transform
Step 8: imageCompression (ZFAry, skeletonAry, skeletonFile, outFile1) // Perform lossless compression
Step 9: close skeletonFile
Step 10: reopen skeletonFile
Step 11: setZero (ZFAry)
Step 12: loadSkeleton (skeletonFile, ZFAry)
Step 13: imageDeCompression (ZFAry, outFile1) // Perform decompression
Step 14: deCompressFile ← output numRows, numCols, newMinVal, newMaxVal
Step 15: threshold (ZFAry, deCompressFile,1)
Step 16: close all files

****
V. Compute8Distance (ZFAry, outFile1)
****

Step 1: fistPass8Distance (ZFAry)
Step 2: reformatPrettyPrint (ZFAry, outFile1) // with proper caption i.e., 1st pass distance transform
Step 3: secondPass8Distance (ZFAry)
Step 4: reformatPrettyPrint (ZFAry, outFile1) // with proper caption i.e., 2nd pass distance transform

****

VI. imageCompression (ZFAry, skeletonAry, skeletonFile, outFile1)

****

Step 1: computeLocalMaxima (ZFAry, skeletonAry)
Step 2: reformatPrettyPrint (skeletonAry, outFile1)// with proper caption i.e., Local maxima, skeletonAry
Step 3: extractSkeleton (skeletonAry, skeletonFile)

****

VII. imageDeCompression (ZFAry, outFile1)

****

Step 1: firstPassExpension (ZFAry)
Step 2: reformatPrettyPrint (ZFAry, outFile1) // with proper caption i.e., 1st pass Expansion
Step 3: secondPassExpension (ZFAry)
Step 4: reformatPrettyPrint (ZFAry, outFile1) // with proper caption i.e., 2nd pass Expansion