Project 1 (in Java): Given a bimodal histogram of a grey-scale image, you are to implement one of the two automatic threshold selections: the bi-Gaussian method.

****

Language: Java

****

Project name: Bi-Means automatic threshold selection

Project points: 10 pts

Due Date: <u>Soft copy (*.zip) and hard copies (*.pdf)</u>:

+1 (11/10 pts): early submission, 9/10/2023, Sunday before midnight

(10/10 pts): on time, 9/13/2023 Wednesday before midnight

(-10/10 pts): non-submission, 9/13/2023 Wednesday after midnight

*** Name your soft copy and hard copy files using the naming convention given in the project submission requirement.

*** All submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in **<u>the same email attachments</u>** with correct email subject as stated in the project submission requirement, otherwise, your submission will be rejected.

*** Inside the email body includes:

- Your answer to the five questions states in the project submission requirements.
- Screen recoding link. (See the Screen Recording requirements.)

*** You must open all files via args[], -5 points if hard-code the file names.

==

You are given histograms: histogram1 and histogram2.

What you need to do:

1. Implement your program as given the specs below.
2. Run your program twice: once using histogram1 and once using histogram2
3. Include in your hard copy *.pdf file as follows:
   - Cover page.
   - Source code.
   - outFile1 for histogram1.
   - deBugFile for histogram1
   - outFile1 for histogram2.
   - deBugFile for histogram2.

****

I. Inputs:

****

a) inFile1 (args [0]): a text file representing a histogram of a gray-scale image. The input format as follows:

```
For example:
5  7  0  9     // 5 rows, 6 cols, min is 0 max 9
0  2           // hist [0] is 2
1  8           // hist [1] is 8
2  5           :
:
```

****

II. Outputs:

****

a) outFile1 (args [1]): including the following 2 items: i) and ii):

i) A 2-D display of the histogram (for visual)

```
// use  small font size 2 or 3 or 4 so that the longer +++++++++++++++++ can fit in the width of a page.
4 6 1 12    // image header
0  (0):
1  (2):++
2  (3):+++
3  (5):++++++
4  (10):++++++++++
5  (12):++++++++++++
```

6 (10):++++++++++
7 (8):++++++++
8 (6):++++++
9 (6):++++++
10 (4):++++
11 (2):++
12 (1):+

ii) The Bi-Gaussian auto-selected threshold value. // with caption.

b) deBugFile (args [2]): For all debugging prints.

****
III, Data structure:
****

- a thresholdSelection class
  - (int) numRows, numCols, minVal, maxVal
  - (int []) histAry// a 1D integer array (size of maxVal + 1) to store the histogram.
    // It needs to be dynamically allocated at run time; **initialize to zero**.
  - (int []) GaussAry // a 1D integer array (size of maxVal + 1) to store the "modified" Gaussian function.
    // It needs to be dynamically allocated at run time.
  - (int) BiGaussThrVal // the auto selected threshold value by the Bi-Gaussian method.
  - (int) maxHeight // The largest hist[i] within a given range of the histogram.

  Methods:
  - constructor (…) // It dynamically allocates all member arrays and initialization.
  - (int) loadHist (…) // reads and loads the histAry from inFile and **returns** the max hist[i]. // On your own
  - dispHist (histAry, outFile1) // Display the histogram in the format as shown in the above. // On your own.
  - setZero (Ary) // Set 1D Ary to zero; //on your own.
  - (int) biGauss (…) // See algorithm below.
    // The method determines the best threshold selection (via fitGauss method)
    // where the two Gaussian curves fit the histogram the best.
  - (double) computeMean (…) // See algorithm below.
    // Computes the mean from leftIndex to rightIndex of the histogram.
    // and returns the *weighted* average of the histogram; i.e., i * hist[i].
  - (double) computeVar (…) // See algorithm below. Computes the *weighted* variance from the given leftIndex
    // to rightIndex of the histogram and returns the *weighted* variance.
  - modifiedGauss (x, mean, var, maxHeight)
    // The original Gaussian function is
    // $g(x) = a* \exp (- ((x-b)^2)/(2*c^2)))$
    // where a is the height of the Gaussian Bell curve, i.e.,
    // $a = 1/(\mathrm{sqrt}(c^2 * 2 * pi)$; b is mean and c^2 is variance
    // Here, the modified method replace 'a' in g(x) with maxHeight of histograma
    // $G(x) = maxHeight * \exp(- ((x-mean)^2 / (2* c^2))$
    // The method returns G(x)
    // Alternatively, instead of using maxHeight, one can use
    // G(x) = maxHeight / maxGVal * g(x), where
    // maxGVal is the largest g(x).
    // If you are interest, you may use as such,
    // however, use maxHeight is good enough for this project.
  - fitGauss (...) // computes the Gaussian curve fitting to the histogram; see algorithm below

****

IV. Main (...) // copy the algorithm steps from step 0 to step 4 below and past to the "cover page" of your pdf.

****

Step 0: inFile1, outFile1, deBugFile ← open via args []
Step 1: numRows, numCols, minVal, maxVal ← read from inFile1.
    histAry ← dynamically allocate (size of maxVal + 1) and initialized to zero.
    maxHeight ← loadHist (histAry, inFile) // loadHist ( ) returns the largest value of histogram.
    dynamically allocate all other arrays and initialized to zero.
Step 2: dispHist (histAry)
Step 3: BiGaussThrVal ← biGaussian (histAry, GaussAry, maxHeight, minVal, maxVal, deBugFile)
    outFile1 ← output BiGaussThrVal with caption
Step 4: close all files

****

V. (int) biGaussian (histAry, GaussAry, maxHeight, minVal, maxVal, deBugFile)

****

Step 0: deBugFile ← output "Entering biGaussian method" // debug print
    (double) sum1
    (double) sum2
    (double) total
    (double) minSumDiff
    offSet ← (int) (maxVal - minVal) / 10
    dividePt ← offSet
    bestThr ← dividePt
    minSumDiff ← 99999.0 // a large value
Step 1: setZero (GaussAry) // reset in each iteration
step 2: sum1 ← fitGauss (0, dividePt, histAry, GaussAry, deBugFile) // fitting the first Gaussian curve
Step 3: sum2 ← fitGauss (dividePt, maxVal, histAry, GaussAry, deBugFile) // fit the second Gaussian curve
Step 4: total ← sum1 + sum2
Step 5: if total < minSumDiff
        minSumDiff ← total
        bestThr ← dividePt
Step 6: deBugFile ← print dividePt, sum1, sum2, total, minSumDiff and bestThr
Step 7: dividePt ++
step 8: repeat step 1 to step 9 while dividePt < (maxVal – offSet)
Step 9: deBugFile ← "leaving biGaussian method, minSumDiff = bestThr is " print minSumDiff and bestThr
step 10: return bestThr

****

V. double fitGauss (leftIndex, rightIndex, histAry, GaussAry, deBugFile)

****

Step 0: deBugFile ← output "Entering fitGauss method" // debug print
    (double) mean
    (double) var
    (double) sum ← 0.0
    (double) Gval
    (double) maxGval
step 1: mean ← computeMean (leftIndex, rightIndex, maxHeight, histAry, deBugFile)
    var ← computeVar (leftIndex, rightIndex, mean, histAry, deBugFile, deBugFile)
Step 2: index ← leftIndex
Step 3: Gval ← modifiedGauss (index, mean, var, maxHeight) // see equation below.
Step 4: sum += abs (Gval – (double)histAry[index])
Step 5: GaussAry[index] ← (int) Gval
Step 6: index ++
Step 7: repeat step 3 – step 6 while index <= rightIndex
Step 8: deBugFile ← "leaving fitGauss method, sum is;" print sum // debug print
Step 9: return sum

```
************************************
VI. (double) computeMean (leftIndex, rightIndex, maxHeight, histAry, deBugFile)
************************************
Step 0: deBugFile ← output "Entering computeMean method" // debug print
        maxHeight ← 0 // maxHeight came via parameter, it is a reference variable, NOT local variable!
        sum ← 0
        numPixels ← 0
Step 1: index ← leftIndex
Step 2: sum += (hist[index] * index)
       numPixels += hist[index]
Step 3: if hist[index] > maxHeight
                maxHeight ← hist[index]
Step 4: index++
Step 5: repeat Step 2 to step 4 while index < rightIndex
Step 6: (double) result ← (double) sum / (double) numPixels
Step 7: deBugFile ← output "Leaving computeMean method maxHeight is an result " print  maxHeight and result
Step 8: return result
************************************
V. double fitGauss (leftIndex, rightIndex, histAry, GaussAry, deBugFile)
************************************
Step 0: deBugFile ← output "Entering fitGauss method" // debug print
        (double) mean
        (double) var
        (double) sum ← 0.0
        (double) Gval
        (double) maxGval
step 1: mean ←  computeMean (leftIndex, rightIndex, maxHeight, histAry, deBugFile)
        var ← computeVar (leftIndex, rightIndex, mean, histAry, deBugFile, deBugFile)
Step 2: index ← leftIndex
Step 3: Gval ← modifiedGauss (index, mean, var, maxHeight) // see equation below.
Step 4: sum += abs (Gval – (double)histAry[index])
Step 5: GaussAry[index] ← (int) Gval
Step 6: index ++
Step 7: repeat step 3 – step 6 while index <= rightIndex
Step 8: deBugFile ← "leaving fitGauss method, sum is;" print sum // debug print
Step 9: return sum

************************************
IV. (double) computeVar (leftIndex, rightIndex, mean, histAry, deBugFile, deBugFile)
************************************
Step 0: deBugFile ← output "Entering computeVar  method" // debug print
        sum ← 0.0
        numPixels ← 0
Step 1: index ← leftIndex
Step 2: sum += (double) hist [index] * ((double) index – mean)^2)
       numPixels += hist[index]
Step 3: index++
Step 4: repeat Step 2 to step 3 while index < rightIndex
Step 5: (double) result ← sum / (double) numPixels
Step 6: deBugFile ← output "Leaving computeVar method returning result " print result // debug print
Step 7: return result
************************************
X. (double) modifiedGauss (x, mean, var, maxHeight)
************************************
return (double) (maxHeight * exp ( - ( ((double) x-mean)^2 / (2*var) )
                // double check the equation with the equation given in class!!
```