Project 2 (Java): Bi-Means auto threshold selection. You are to implement the Bi-Means auto threshold selection method taught in class.

What you need to do:
1. Implement your program with respect to the specs given below and debug your program until your program compiles.
2. You will be given two histogram data files: hist1 and hist2.
3. Run your program twice: once with hist1 and once with hist2.
4. Before you submit, pay attention to all your outputs to see if all outputs fit within the pages. No wrapped around.

*** Include in your hard copy *PDF.pdf file as follows:
- Cover page.
- source code.
- Output histFile for hist1.
- Output GaussFile for hist1.
- Output logFile for hist1. // limit to 3 pages if more.
- Output histFile for hist2.
- Output GaussFile for hist2.
- Output logFile for hist2. // limit to 3 pages if more.

****

Language: Java

****

Project name: Bi-Means automatic threshold selection
Project points: 10 pts
Due Date: Soft copy (*.zip) and hard copies (*.pdf):
+1 (11/10 pts): early submission, 9/15/2024, Sunday before midnight (11:59pm)
(10/10 pts): on time, 9/19/2024, Thursday before midnight
(-10/10 pts): non-submission, 9/19/2024. Thursday after midnight

*** Name your soft copy and hard copy files using the naming convention given in Project Submission Requirements.

*** All on-line submission MUST include Soft copy (*.zip) and hard copy (*.pdf) in the same email attachments with correct email subject as below; otherwise, your submission will be rejected.
*** Inside the email body include your answer to the 4 questions. **Optional screen recording if you wish**.

****

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) histFile (args [1]): This file includes the followings:
i) Print the input data as the same format as printHist in your project 1

ii) A 2-D display of the histogram with image header in the same format as dispHist in your project 1.

b) GaussFile (args [2]): This file includes the follows:
i) Print the selected threshold value.

ii) Print the bestFitGaussAry array (similar to histogram).

iii) A 2-D display of the best fitted Gaussians (see dispGaussGraph method below.)

iv) A 2-D display of the gaps between the histogram and the two best-fitted Gaussian curves.

c) logFile (args [3]): To log the progress of your program.

****

III, Data structure:

****

- BiMeans class
  - (int) numRows, numCols, minVal, maxVal // image header.
  - (int) BiGaussThrVal // the auto selected threshold value by the Bi-Gaussian method.
  - (int) histHeight // The largest hist[i] in the input histogram.
  - (int) maxHeight // The largest hist[i] within a given range of the histogram. **Initialize to 0.**
  - (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 curve values.
    // It needs to be dynamically allocated at run time. **initialize to zero**.
  - (int []) bestFitGaussAry // to store the best biGaussian curves. **Initialize to zero**.
  - (char [][]) GaussGraph // a 2-D char array size of maxVal+1 by histHeight+1, **initialize to blanks**,
  - (char [][]) gapGraph // a 2-D char array size of maxVal+1 by histHeight+1, **initialize to blanks.**

  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].
  - printHist (…) // Output histAry to histFile. Re-use code from project 1.
  - dispHist (…) // Graphic displays histAry to histFile. Re-use code from project 1
  - copyArys (ary1, ary2) // copy ary1 to ary2.
  - setZero (Ary) // Set 1D Ary to zero.
  - setBlanks (graph) // Set graph to blank " ".
  - (int) biGaussian (…) **// See algorithm below**.
    // The method determines the best threshold selection (via fitGauss method)
    // where the two Gaussian curves fit the histogram the best.
  - fitGauss (...) // computes the Gaussian curve fitting to the histogram; **see algorithm below.**
  - (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 (…) // Computes the *weighted* variance from the given leftIndex
    // to rightIndex of the histogram and returns the *weighted* variance. **See algorithm below**.
  - modifiedGauss (x, mean, var, maxHeight) **// See algorithm below.**
  - printBestFitGauss (…) // Output bestFitGaussAry to GaussFile, in the format similar to printHist.
    //(Don't forget image header).
  - plotGaussGraph (…) // plot the bestFitGaussAry onto GaussGraph with '*'. **See algorithm below.**
  - dispGaussGraph (…) // Output GaussGraph to GaussFile. (Don't forget image header).
  - plotGapGraph (…) // plot the gaps between the histogram and the best fitted Gaussian curves with "@".
    // **See algorithm below**.
  - dispGapGraph (…) // Output gapGraph to GaussFile. (Don't forget image header).

****

IV. Main (...)

****

```
Step 0: check if argc count is correct and each file can be opened.
        inFile1, histFile, GaussFile, logFile ← open via args []
Step 1:  numRows, numCols, minVal, maxVal ← read from inFile1.
         histHeight ← loadHist (histAry, inFile) // loadHist ( ) returns the largest value of  input histogram.
         dynamically allocate all arrays with proper size and proper initializations.
Step 2: histFile ← "** Below is the input histogram **")
        printHist (histAry, histFile)
        histFile ← "** Below is the graphic display of the input histogram **".)
        dispHist (histAry, histFile)

Step 3: BiGaussThrVal ← biGaussian (histAry, GaussAry, maxHeight, minVal, maxVal, Graph, logFile)
        GaussFile ← "** The selected threshold value is " // print the value
```

```
Step 4: GaussFile ← "** Below is the best Fitted Gaussians **"
        printBestFitGauss (bestFitGaussAry, GaussFile)
Step 5: plotGaussGraph (bestFitGaussAry, GaussGraph, logFile)
        GaussFile ← "** Below is the graphic display of BestFitGaussAry **".
        dispGaussGraph (GaussGraph, GaussFile) // Output GaussGraph to GaussFile.
Step 6:  plotGapGraph (histAry, bestFitGaussAry, gapGraph, logFile)
        GaussFile ← "** Below displays the gaps between the histogram and the best fitted Gaussians **"
        dispGapGraph (gapGraph, GaussFile) // Output gapGraph to GaussFile.
Step 7: close all files
************************************
V. (int) biGaussian (histAry, GaussAry, maxHeight, minVal, maxVal, Graph, logFile)
************************************
Step 0: logFile ← output "Entering biGaussian method"
        (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, maxHeight, Graph, logFile) // first Gaussian curve
Step 3: sum2 ← fitGauss (dividePt, maxVal, histAry, GaussAry, maxHeight, Graph, logFile) //second Gaussian curve
Step 4: total ← sum1 + sum2
Step 5: if total < minSumDiff
                minSumDiff ← total
                bestThr ← dividePt
               copyArys (GaussAry, bestFitGaussAry)

Step 6: logFile ← "In biGaussian (): dividePt = , sum1= , sum2= , total= , minSumDiff = and bestThr="
                                    //print those values.
Step 7: dividePt ++
Step 8: repeat step 1 to step 7 while dividePt < (maxVal – offSet)

Step 9:  logFile ← "leaving biGaussian method, minSumDiff =   bestThr is " print minSumDiff and bestThr
step 10: return bestThr

************************************
VI. (double) fitGauss (leftIndex, rightIndex, histAry, GaussAry, maxHeight, Graph, logFile)
************************************
Step 0: logFile ← "Entering fitGauss method"
        (double) mean
        (double) var
        (double) sum ← 0.0
        (double) Gval
Step 1: mean ←  computeMean (leftIndex, rightIndex, maxHeight, histAry, logFile)
        var ← computeVar (leftIndex, rightIndex, mean, histAry, logFile)
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: logFile ← "leaving fitGauss method, sum is;" print sum // debug print
Step 9: return sum
```

```
************************************
VII. (double) computeMean (leftIndex, rightIndex, maxHeight, histAry, logFile)
************************************
Step 0: logFile ← output "Entering computeMean method"
        maxHeight ← 0 // maxHeight came via parameter, it is a reference variable, NOT local variable!
                      // If you like, maxHeight need NOT passes in the parameter, just use it as global 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: logFile ← output "Leaving computeMean method maxHeight =  result = " print  maxHeight and result
Step 8: return result
```

```
************************************
VIII. (double) computeVar (leftIndex, rightIndex, mean, histAry, logFile)
************************************
Step 0: logFile ← output "Entering computeVar() method"
        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: logFile ← output "Leaving computeVar method returning result " print result
Step 7: return result
```

```
*********************************************
IX. plotGaussGraph (bestFitGaussAry, GaussGraph, logFile)
**********************************************
Step 0: logFile ← "Entering plotGaussGraph () method"
        setBlanks (GaussGraph)
Step 1: index ← 0
Step 2: if bestFitGaussAry [index] > 0
Step 3: i ← 0
Step 4: GaussGraph [index][i] ← "*"
Step 5: i++
Step 6: repeat Step 4 to 5 while i < bestFitGaussAry [index]
Step 7: index++
Step 8: repeat Step 2 to Step 7 while index <= maxVal
Step 9: logFile ← "Leaving plotGaussGraph () method"
```

****

X. plotGapGraph (histAry, bestFitGaussAry, logFile)

****

```
Step 0: logFile ← "Entering plotGapGraph () method"
        setBlanks (GaussGraph)
Step 1: index ← 0
Step 2: if bestFitGaussAry [index] <= histAry[index]
                end1 ←  bestFitGaussAry [index]
                end2 ← histAry[index]
        else
                end1 ← histAry[index]
                end2 ← bestFitGaussAry [index]
Step 3: i ← end1
Step 4: Graph[index][i] ← '@'
Step 5: i++
Step 6: repeat Step 4 to Step 5 while i <= end2
Step 7:  index++
Step 8: repeat Step 2 to Step 7 while index <= maxVal
Step 9: logFile ← "leaving plotGapGraph ()"
```

****

XI. (double) modifiedGauss (x, mean, var, maxHeight)

****

```
return (double) (maxHeight * exp ( - ( ( (double) x) -mean)^2 / (2*var) )
                // equation: G(x) = maxHeight * exp (- ( (x-mean)$^2$ / (2* var) ))

    // Note:
    //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/(sqrt(c$^2$ * 2 * pi);  b is mean, c is, σ, the standard deviation and c$^2$ is variance
    // Here, the modified method replace 'a' in g(x) with maxHeight of histogram.
    // 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. The equation:
    // G(x) = maxHeight * exp( - ( (x-mean)$^2$ / (2* var) ) )
```