Project 1 (in C++): 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: C++

****

Project name: Bi-Means automatic threshold selection

Project points: 12 pts

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

+1 (13/12 pts): early submission, 2/17/2024, Saturday before midnight

(12/12 pts): on time, 2/20/2024 Tuesday before midnight

(-12/12 pts): non-submission, 2/20/2024 Tuesday 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.

Email subject: (CV) firstName lastName <Project 1: Bi-Means automatic threshold selection (C++)>

*** Inside the email body includes:

- Your answer to the five questions given at the end of the project submission requirements.
- Screen recoding link. (-2 without the recording!)

*** Place your screen recording in your project submission email body below the 5 questions.

==

You are given histograms: histogram1 and histogram2 to test your program.

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 and outFile2 for histogram1.
   - deBugFile for histogram1 // limit to 4 pages.
   - outFile1 and outFile2 for histogram2.
   - deBugFile for histogram2 // limit to 4 pages.

****

I. Inputs:

****

a) inFile1 (argv [1]): 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: include outFile1, outFile2 and deBugFile.

****

a) outFile1 (argv [2]): This file includes the followings:

i) A 2-D display of the histogram with image header in the format as given below with proper caption; use fixed font (Currier New with size 3 or 4) so that the longest +++++++++++++++++ can fit in the width of a page.

```
4 6 1 10   // image header
0  (0):
1  (2):++
2  (3):+++
3  (5):++++++
4  (10):++++++++++
```

5 (12):++++++++++++
6 (10):++++++++++
7 (8):++++++++
:

b) outFile2 (argv[3]):
  i) The Bi-Gaussian auto-selected threshold value with caption.
  ii) A 2-D display of the graph that shows: the histogram with '+', the two Gaussian best-fitted curve with '*' and the gap points with '^' between the Gaussian curves and the histogram.

c) deBugFile (argv [4]): For all debugging prints as program dictates.

****

III, Data structure:

****

- a thresholdSelection class
  - (int) numRows, numCols, minVal, maxVal // image header.
  - (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.
  - (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**.
  - (char **) Graph // a 2-D char array size of maxVal+1 by maxHeight+1, **initialize to blank**,
    // It needs to be dynamically allocated at run time. Within Graph [] [] use:
      + for histogram points.
      * for the two best-fitted Gaussian curves points.
      ^ for points of gaps in between histogram and the two curves.

  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 (…) // Output the histogram in the format as shown in the above. // On your own.
  - plotHist (…) // plot the histogram onto Graph with '+'. // On your own.
  - plotGraph (row, end1, end2, symbol)
    // plot the given symbol onto graph at row, from end1 to end2. // On your own.
  - setZero (Ary) // Set 1D Ary to zero; //on your own.
  - (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]. **See algorithm below**.
  - (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)
    // 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.,
    // $a = 1/(\text{sqrt}(c^2 * 2 * pi)$; 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:

    // $G(x) = maxHeight * \exp(- ((x-mean)^2 / (2* var))$

****

IV. Main (...)

****

Step 0: inFile1, outFile1, outFile2, deBugFile ← open via args []
Step 1: numRows, numCols, minVal, maxVal ← read from inFile1.
    maxHeight ← loadHist (histAry, inFile) // loadHist ( ) returns the largest value of histogram.
    dynamically allocate histAry, GaussAry, Graph with proper size and proper initializations.
Step 2: outFile1 ← "in main(), below is the input histogram"
    dispHist (histAry, outFile1)
Step 3: plotHist (histAry, Graph)
    deBugFile ← "In main(), below is the Graph after plotting the histogram onto Graph"
    deBugFile ← print the Graph to deBugFile.
Step 4: BiGaussThrVal ← biGaussian (histAry, GaussAry, maxHeight, minVal, maxVal, Graph, deBugFile)
Step 5: outFile2 ← "The BiGaussThrVal is" print BiGaussThrVal
Step 6: outFile2 ← "In main(). Below is the graph showing the histogram, the best fitted Gaussian curves and the gap"
Step 7: outFile2 ← "In main(), Below is the final Graph"
    outFile2 ← output Graph
Step 8: close all files

****

V. (int) biGaussian (histAry, GaussAry, maxHeight, minVal, maxVal, Graph, 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, maxHeight, Graph, deBugFile) // first Gaussian curve
Step 3: sum2 ← fitGauss (dividePt, maxVal, histAry, GaussAry, maxHeight, Graph, deBugFile) //second Gaussian curve
Step 4: total ← sum1 + sum2
Step 5: if total < minSumDiff
        minSumDiff ← total
        bestThr ← dividePt

Step 6: deBugFile ← "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: deBugFile ← "leaving biGaussian method, minSumDiff = bestThr is " print minSumDiff and bestThr
step 10: return bestThr

****

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

****

Step 0: deBugFile ← "Entering fitGauss method" // debug print
    (double) mean
    (double) var
    (double) sum ← 0.0
    (double) Gval
Step 1: mean ← computeMean (leftIndex, rightIndex, maxHeight, histAry, deBugFile)
    var ← computeVar (leftIndex, rightIndex, mean, histAry, 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: Graph[index][(int) Gval] ← '*'
Step 7: if (int) Gval <= histAry[index]
end1 ← (int) Gval
end2 ← histAry[index]
else
end1 ← histAry[index]
end2 ← (int) Gval
Step 8: plotGraph (index, end1, end2, '^')
Step 9: index ++
Step 10: repeat step 3 – step 9 while index <= rightIndex
Step 11: deBugFile ← "leaving fitGauss method, sum is;" print sum // debug print
Step 12: 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!
// 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: deBugFile ← output "Leaving computeMean method maxHeight is an result " print maxHeight and result
Step 8: return result
****
IV. (double) computeVar (leftIndex, rightIndex, mean, histAry, 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
Step 7: return result

****
X. (double) modifiedGauss (x, mean, var, maxHeight)
****
return (double) (maxHeight * exp (- (((double) x) -mean)^2 / (2*var))
// equation: $G(x) = maxHeight * \exp(- ((x-mean)^2 / (2* var))$