```
// include header files
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cfloat>
#include <cmath>
#include <gl/glew.h>
#include <GL/glut.h>
#include "utility.h"

using namespace std;

// File names
char* fileName1, *fileName2;
string currentFile;

// Input data
float* dataset;
int numDataPoints;
float minimum, maximum;

// Histogram
int numIntervals = 30;
float* endPoints;
float* prob;
float maxProb = -1;

// Theoretical distributions
int curveType = 0;
int numCurvePoints = 100;
float* curveX = new float[numCurvePoints];
float* curveY = new float[numCurvePoints];

// Parameters
float mu = 0, sigma = 1;  // Normal distribution
float lamda = 1;          // Exponential distribution
float parameterStep = 0.05;        // Step size for changing parameter values

// Drawing parameters
int width = 800, height = 600;
float world_x_min, world_x_max, world_y_min, world_y_max;
float axis_x_min, axis_x_max, axis_y_min, axis_y_max;

// Compute all the points for normal distribution
void computeNormalFunc(float mu, float sigma)
{
       // Determine the step size and compute the arrays curveX and curveY
       // (numCurvePoints).
}

// Compute all the points for exponential distribution
void computeExponentialFunc(float lamda)
{
       // Determine the step size and compute the arrays curveX and curveY
       // (numCurvePoints).
}
```

```
void display(void)
{
	/* clear all pixels  */
	glClear (GL_COLOR_BUFFER_BIT);

	// Reset Modelview matrix.
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();

	glLineWidth(1);
	glColor3f(1, 1, 1);

	// Draw x and y axes

	// Display the maximum probability value

	// Draw probability histogram

	// Draw the theoretical distribution using thicker lines.

	// Compute the top-left position of the annotation

	// File Name

	// Minimum

	// Maximum

	// Number of intervals

	// Draw theoretical distributions

	glFlush ();
	glutSwapBuffers();
}

void init (void)
{
	glClearColor (0.0, 0.0, 0.0, 0.0);
	glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
}

// Compute the probability for the histogram (vertical axis)
void computeProbability(int numIntervals)
{
	// Delete previously allocated memory

	// Determine the end points for each interval (update the array endPoints).

	// Re-initialize the maximum probability after the number of intervals has been
	// changed.

	// Compute the probability for each interval (update the array prob).
}

void readFile(string fileName)
{
	ifstream inFile(fileName);
```

```
    if (!inFile.is_open())
    {
        cout << fileName << " couldn't be opened.\n";
        system("pause");
        exit(1);
    }

    inFile >> numDataPoints;

    // Memory management.
    if (dataset != NULL)
        delete data;
    dataset = new float[numDataPoints];

    minimum = FLT_MAX;
    maximum = -FLT_MAX;

    // Read the data and compute minimum and maximum.
    for (int i = 0; i < numDataPoints; i++)
    {
        inFile >> dataset[i];
        if (dataset[i] < minimum)
            minimum = data[i];
        if (dataset[i] > maximum)
            maximum = dataset[i];
    }

    // Compute the limits for the axes and world.

    // Compute the histogram

    // Compute the theoretical distribution.
}

void keyboard(unsigned char key, int x, int y)
{
    if (key == 'q' || key == 'Q' || key == 27)
        exit(0);
}

void specialKey(int key, int x, int y) // for the arrow keys
{
    // Update the parameters and theoretical distributions
    glutPostRedisplay();
}

void topMenuFunc(int id)
{
    exit(0);
}

void fileMenuFunction(int id)
{
    // Read file.

    // Upadte projection since the data has changed.
```

```
        glutPostRedisplay();
}

void funcMenuFunction(int id)
{
        // Different theoretical distributions
}

void histogramMenuFunction(int id)
{
        // Update the number of intervals and recomputed the histogram.

        // Update projection since the histogram has changed due to the change of number
        // of bars.
        glutPostRedisplay();
}

void parameterStepMenuFunction(int id)
{
        // Update the parameter step size.
}

void createMenu()
{
        // Create menus
}

void reshape(int w, int h)
{
        // Set matrix mode and projection.
}

int main(int argc, char** argv)
{
        // Program initialization
        // GLUT initialization
        // Set up callbacks.
        // Enter the GLUT main loop.
}
```