DSP2 SS2020 – Exercise 3: Histogram and Point Operations

1. Relevant source files:
    - main.cpp: includes the main function and the program structure
    - Histogram.h: declaration of the histogram class
    - Histogram.cpp: implementation of the histogram class
    - PointOperations.h: declaration of the point operations class
    - PointOperations.cpp: implementation of the point operations class
    - Threshold.h: declaration of the threshold class
    - Threshold.cpp: implementation of the threshold class

2. Histogram.cpp:
    a. Implement the histogram computation in the function "void Histogram::calcHist(const cv::Mat &input, cv::Mat &hist)" :
        - "Input" is the input image
        - "hist" is for the result of your implementation
    b. Implement the "void Histogram::calcStats(const cv::Mat &hist, uchar &min, uchar &max, uchar &mean)" function (calculate the minimum, maximum and mean value of the image):
        - hist: input histogram (read the values from this variable)
        - min: result for the minimum
        - max: result for the maximum
        - mean: result for the mean value

3. PointOperations.cpp:
    a. Implement the function "void PointOperations::adjustContrast(cv::Mat &input, cv::Mat &output, float alpha, uchar center)" for contrast adjustment:
        - input: input image
        - output: output image for the result
        - alpha: value for the contrast adjustment
        - center: center point for the contrast adjustment
    b. Implement the function "void PointOperations::adjustBrightness(cv::Mat &input, cv::Mat &output, int alpha)" for the brightness adjustment:
        - input: input image
        - output: output image for the result
        - alpha: value for the brightness adjustment
    c. Implement the function "void PointOperations::invert(cv::Mat &input, cv::Mat &output)" for the image inversion:
        - input: input image
        - output: output image for the result
    d. Implement the function "void PointOperations::quantize(cv::Mat &input, cv::Mat &output, uchar n)" for the image quantization:
        - input: input image
        - output: output image for the result
        - n: number of quantization bits for the output image
        - note: the output image also has 8 bits, think about the bit positions in your "uchar" for a correct image representation.