In this assignment, you are required to implement several different edge detection algorithms. You can reuse the code from your previous assignments (e.g., image I/O and filtering). For simplicity, you can also assume that the input images are gray-scale ones.

**Requirement**

You should implement the following functions:

- **Sobel/Prewitt edge detection**: Filter the input image with the 3x3 Sobel or Prewitt operators, compute the gradient magnitude, and threshold the gradient magnitude. The threshold is inclusive, i.e., using \( \geq \) instead of \( > \) in your code. Save the output as a gray scale image (white (255) for edges). The threshold is supplied as a command line argument. For pixels on the boundary, clamp the x-y coordinate. This rule applies to all filter operations in this assignment.

- **Canny edge detector**: Improve your previous edge detection result using the canny edge detection method. You should follow the steps in [http://fourier.eng.hmc.edu/e161/lectures/canny/node1.html](http://fourier.eng.hmc.edu/e161/lectures/canny/node1.html), except Step 3 (the thresholding of the initial gradient map). This step is only useful to speed up the computation. You can always use the 3x3 Sobel operators for the gradient image. There are three parameters to the algorithm:
  - The standard deviation for Gaussian smooth (\( \sigma \)): This is a floating point number to control the width of the Gaussian function. Typically, the filter size (\( N_0 \)) is \( 2 \times 2 \sigma + 1 \), i.e., if \( \sigma \) is 2, \( N_0 \) is 9x9. The values of the kernel should be filled using the Gaussian function (details can be found in the lecture about spatial filtering). Sigma up to 4 should be supported.
  - Two thresholds for Canny (\( t_1, t_2 \)): Thresholds for edge link, both are floating point number. Note that \( t_1 \leq t_2 \).
  - In order to achieve bit-consistent edge map for grading, the following edge linking procedure should be used. That is, for a given pixel \((x,y)\) whose gradient magnitude \( M(x,y) \geq t_2 \), recursively look for \((x,y)\)'s 8-neighbors whose gradient magnitude \( \geq t_1 \). Repeat the above procedure for all pixels.

- **Command line syntax**:
  ```
  -i inputFileName -o outputFileName {-s t}|{-p t}|{-c sigma t1 t2}
  -i: input image name.
  -o: output image name.
  ```
-s: edge detection with the Sobel operators, t is the threshold value;
-p: edge detection with the Prewitt operators
-c: canny edge detection, parameters as explained above.
The –s, -p,-c switches are mutually exclusive.


Submission
Please follow the general submission guidelines in the first assignment.

Specific to this submission: you should provide one window executable, named as “HW3edge.exe”, in the same directory as the source code. It should be self-contained, i.e., without the need for any DLLs except the DLL for image debugger. It should accept command-line arguments as defined in the requirement section. A set of different test arguments will be used to invoke the executable. The output images will be checked against the standard ones.