The University of Western Ontario
London, Ontario, Canada
Department of Computer Science
CS 4481b/9628b - Image Compression
Assignment 1
Due Thursday February 8, 2018 at 11:55 PM

INDEPENDENT WORK is required on each assignment.

After finishing the assignment, you have to do the following:
- Type your report and convert it to *PDF format*, which must include:
    - Answers to all questions/requirements in the assignment, if any
    - A copy of all programs that you have written
    - A *flowchart* of each function that you have written for **Program-1**, **Program-2**, and **Program-3** sections
    - Images to represent various test cases of your program
    - The caption of each image must:
        - include the parameters that you used to generate the image
        - include the actual image name
        - be stand alone, i.e., there must be enough description of the figure so that a reader should not need to go back to the text to understand the figure

- Prepare a soft-copy submission, including:
    - A copy of your *typed PDF* report
    - All generated images included in your report (use a meaningful image file names)
    - All programs that you wrote

- Upload the soft-copy submission file-by-file, or as an archived directory.

Late assignments are strongly discouraged.
- 10% will be deducted from a late assignment (up to 24 hours after the due date/time)
- After 24 hours from the due date/time, late assignments will not be accepted.

N.B: When marking your assignment, your programs will be tested on the GAUL network. If you will develop your programs in any other platform, make sure that your programs are error free and produce the required outputs when compiling and running them on the GAUL network.

In this assignment, a read and write PBM/PGM/PPM library is provided (included in the attachment as **pnm_library.tar**). Carefully read and understand the code of this library. *Do NOT change anything in the provided library. Just use it.*

Using the provided library, write ONE program *(call it main.c)* to generate and save images as described in **Program-1**, **Program-2**, and **Program-3** sections below.
- Your program must accept *five* arguments *in the command-line*:
    - image type code (e.g.,1 for **pbm**, 2 for **pgm**, or 3 for **ppm**),
    - image width,
    - image height,
    - output image name, and
    - image format code (e.g., 0 for ASCII or 1 for raw).

- When the image type code is 1, i.e., **pbm** image, the content of image is described in **Program-1** below
- When the image type code is 2, i.e., **pgm** image, the content of image is described in **Program-2** below
- When the image type code is 3, i.e., **ppm** image, the content of image is described in **Program-3** below

- The **image width** of **pbm** and **pgm** images *must be* divisible by 4 and greater than or equal to 4.
  If not, you need to display an error message and *stop your program*.
- The **image width** of **ppm** images *must be* divisible by 6 and greater than or equal to 6.
  If not, you need to display an error message and *stop your program*.

- The **image height** of all images *must be* divisible by 4 and greater than or equal to 4.
  If not, you need to display an error message and *stop your program*.

- Your program *must be* able to generate **_square_** images, as well as **_horizontal and vertical rectangle_** images.
- Your program *must* include many inline comments, to facilitate understanding it.

- Include in your submitted report enough *sample runs* to cover various special cases.

- Test cases for each image type (i.e., **pbm**, **pgm**, and **ppm**) *must* include at least a generated horizontal rectangle, a generated vertical rectangle and a generated square.

- For each image type (i.e., **pbm**, **pgm**, and **ppm**), *explain* the content of the generated image when setting:
  o image width to 120 and image height to 4 (*3 images in total*)
  o image width to 4 (6 in case of **ppm** images) and image height to 120 (*3 images in total*).
  Include these generated **pbm**, **pgm**, and **ppm** images as well in your reports as well.

- In your report, write a meaningful caption for each sample output (include in the caption all of the used parameters to generate the image)
- Include a soft-copy of these sample output images with your submitted soft-copy.

In this assignment, a *makefile* file is provided (included in the attachment as makefile) to facilitate the compilation and the testing processes. Carefully read and understand this makefile. *Do NOT change anything in the provided makefile. Just use it.*

FYI: Assignment marking scheme includes, but not limited to,

- In-line commenting
- Error free code on GAUL network (syntax)
- Correct implementation (logically)
- Efficient implementation
- Correctly accepting input from the command line

- Appropriateness of the 3 flowcharts

- The required 120×4, 4×120, and 6×120 images (*6 images in total*) with explanation
- Comprehensive test cases to cover various situations
- The required converted images as stated in **Program-3**

- The neatness of figure captions
- The neatness of the entire report
- The neatness of the written programs
- The neatness of the 3 flowcharts

If your programs are not working properly, you are still encouraged to submit your work for partial mark.
In such case, you should include some comments explaining why your programs are doing so.

Program-1. Draw a flowchart and write a function to generate and save an image similar to the following binary image (pbm).

- o   The white rectangle is located in the middle of the image.
- o   The width of the white rectangle is half the width of the whole image
- o   The height of the white rectangle is half the height of the whole image.
- o   The two diagonal lines must be *solid*, i.e., its neighbouring black pixels must *touch* each other in either one side or one corner.
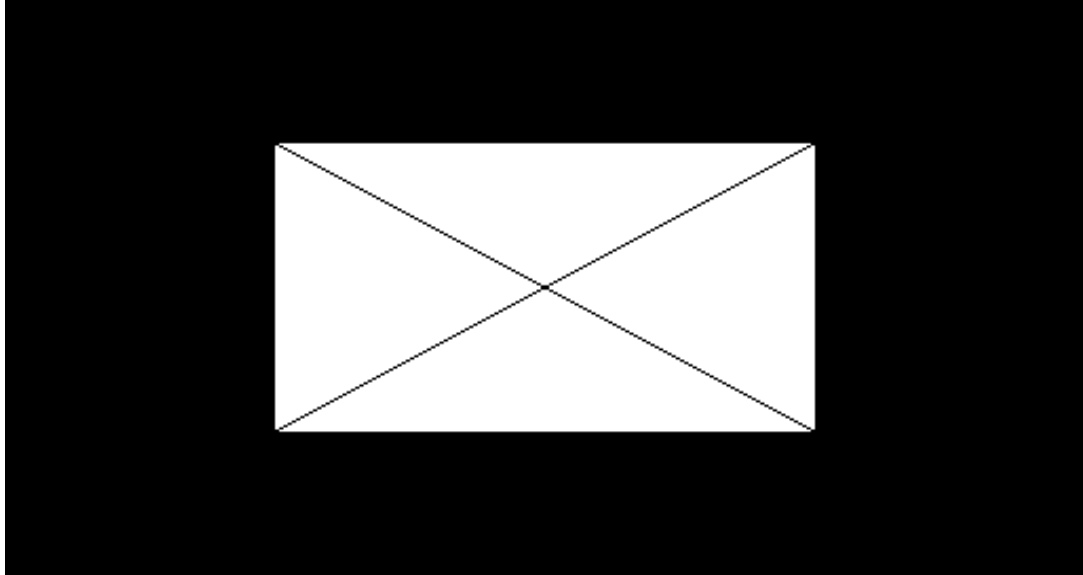
A copy of a PNM ASCII version of the image is included in the attachment list (rectangle_1.pbm)

Program-2. Draw a flowchart and write a function to generate and save an image similar to the following gray image (pgm).
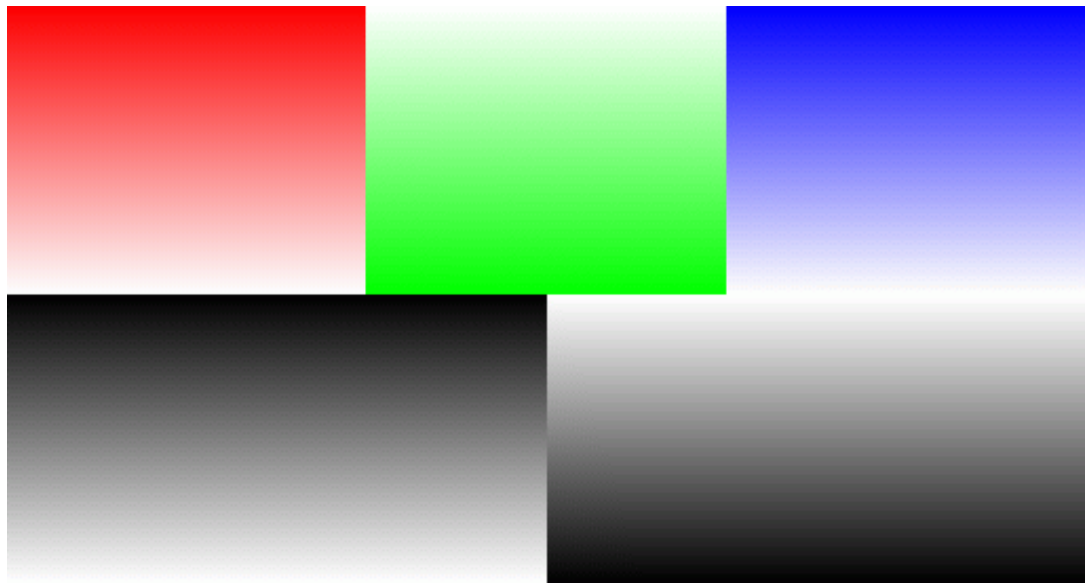
- o   The inner rectangle is located in the middle of the image.
- o   The width of the inner rectangle is half the width of the whole image.
- o   The height of the inner rectangle is half the height of the whole image.
- o   The inner rectangle can be seen as 4 triangles.
- o   The gray level in each horizontal or vertical line in the triangle is the same.
- o   The gray levels in each triangle *gradually change* (line by line) from *white* (at the base of the triangle) to *black* (at the top of the triangle, which is the centre of the image).
- o   Set the maximum_gray_level to 255.
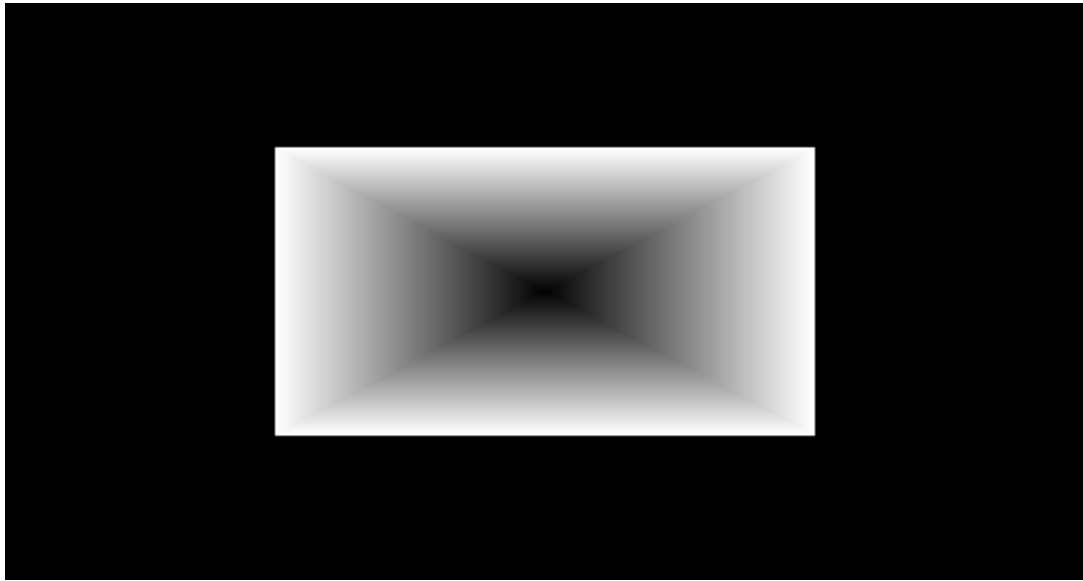
A copy of a PNM ASCII version of the image is included in the attachment list (rectangle_2.pgm)

Program-3. Draw a flowchart and write a function to generate and save an image similar to the following color image (ppm).

- o   The upper half of the image is divided into three parts. The color in each part is *gradually changed* (line by line) from red to white, white to green, and blue to white, respectively.
- o   The lower half of the image is divided into two parts. The color in each part is *gradually changed* (line by line) from black to white and from white to black, respectively.

- o   Set the maximum_gray_level to 255.

- o   Convert and store the generated image into 3 gray images using copy_PPM_to_PGM.
    Use easy to recognize image file names
- o   Comment on the color of each part of the converted images.

A copy of a PNM ASCII version of the image is included in the attachment list (rectangle_3.ppm)