The University of Western Ontario
London, Ontario, Canada
Department of Computer Science
CS 4481b/9628b - Image Compression
Assignment 1
Due Thursday, February 8, 2024, at 11:55 PM

INDEPENDENT WORK is required for 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
  - 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
  - One C program (called *main.c*)
  - All generated images included in your report (use meaningful image file names)
- 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 (compute.gaul.csd.uwo.ca) network. If you develop your programs on 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 (including the extension, i.e., .pbm, .pgm, or .ppm), 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.
- 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:
    - image width to 120 and image height to 4 (*3 images in total*)
    - 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.

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

N.B: In your program, if you use `return 1` to denote that there was an error, the makefile will not continue the rest of the testing cases. To go around this issue, you need to use return 0 instead, but you have to provide an appropriate error message before stopping the program.

To compile your programs, execute "`make all`"
To test the input validation, execute "`make testValidation`"
To test the generated PBM images, execute "`make testPBM`"
To test the generated PGM images, execute "`make testPGM`"
To test the generated PPM images, execute "`make testPPM`"
To test all generated images, execute "`make testAll`"
To delete compiled programs, execute "`make clean`"
To delete all generated images, execute "`make cleanPNM`"

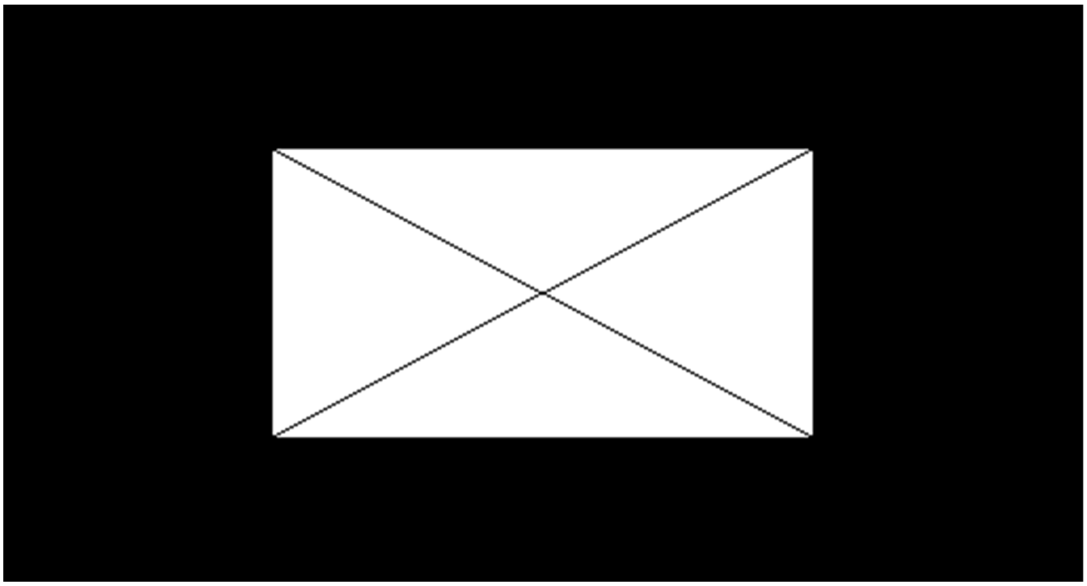
FYI: Assignment marking scheme includes, but is not limited to,

- In-line commenting
- Error-free code on GAUL network (syntax)
- Correct implementation (logically)
- Efficient implementation
- Correctly accepting input from the command line
- The required 120×4, 4×120, and 6×120 images (*6 images in total*) with an 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 program(s)

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

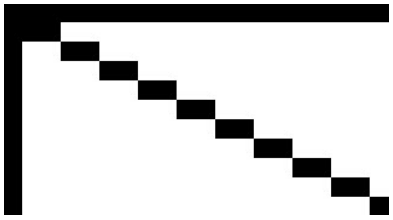
Program-1. Write a ***function*** to generate and save an image similar to the following binary image (pbm).

- The white rectangle is located in the middle of the image.
- The width of the white rectangle is half the width of the whole image
- The height of the white rectangle is half the height of the whole image.
- The two diagonal lines must be ***solid***, i.e., their neighbouring black pixels must ***touch*** each other on either one side or one corner.

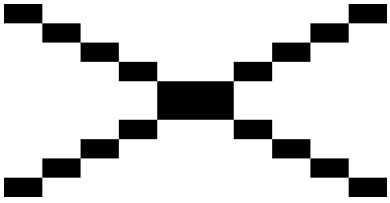


A copy of a PNM ASCII version of the image is included in the attachment list (rectangle_1.pbm)

The following image shows a zoom to the top-left corner of the white rectangle.

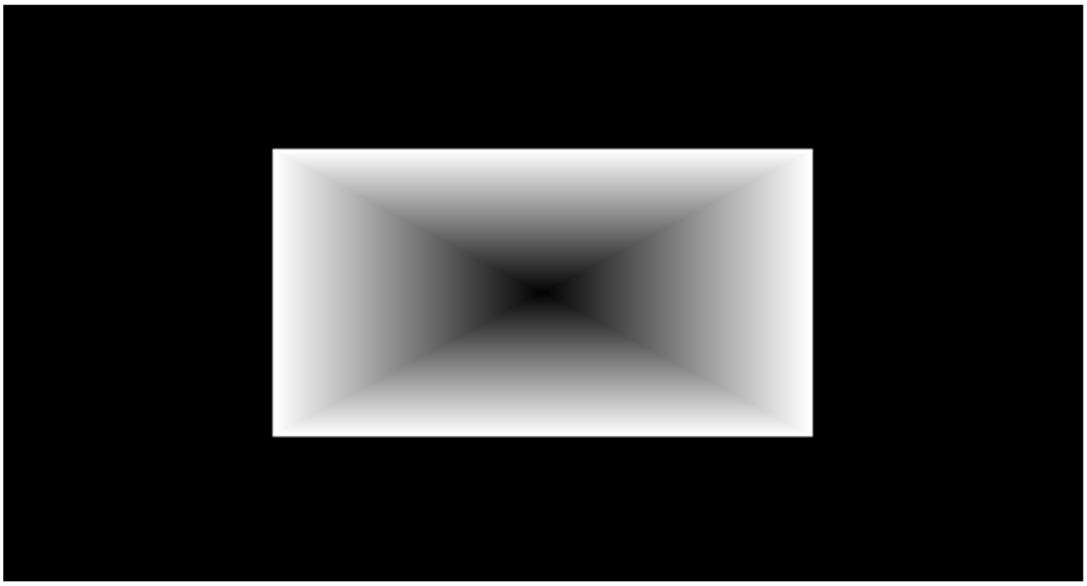


The following image shows a zoom to the centre of the image.



Program-2. Write a ***function*** to generate and save an image similar to the following gray image (pgm).

- The inner rectangle is located in the middle of the image.
- The width of the inner rectangle is half the width of the whole image.
- The height of the inner rectangle is half the height of the whole image.
- The inner rectangle can be seen as 4 triangles.
- The gray level in each horizontal line in the upper and lower triangles or vertical line in the left or right triangles is the same.
- 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).
- Set the maximum_gray_level to 255.



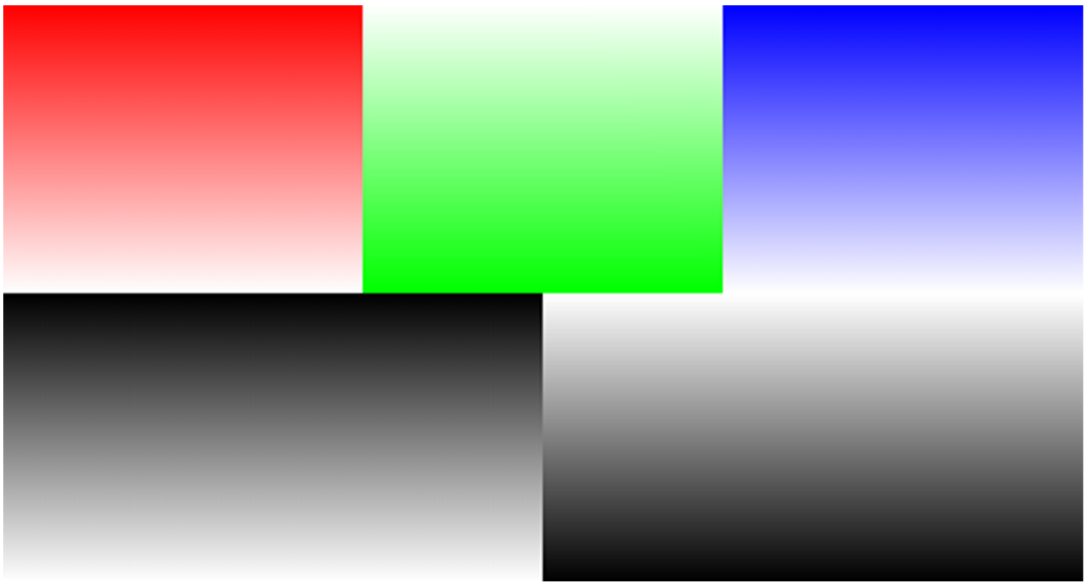
A copy of a PNM ASCII version of the image is included in the attachment list (rectangle_2.pgm)

Program-3. Write a ***function*** to generate and save an image similar to the following color image (ppm).

- 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.
- 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.

- Set the maximum_color_level to 255.

- Convert and store the generated image into 3 gray images using copy_PPM_to_PGM.
  Use easy to recognize image file names
- 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)
