The University of Western Ontario
London, Ontario, Canada
Department of Computer Science
CS 4481b/9628b - Image Compression
Assignment 4
Due Thursday March 30, 2017 at 11:55 PM

INDEPENDENT WORK is required on each assignment.

After finishing the assignment, you have to do the following:

- Type your report, which should include:
  - Answers to all questions/requirements in the assignment
  - A copy of all programs that you have written
- Prepare a soft-copy submission, including:
  - A copy of your *typed* report
  - All programs that you wrote (use meaningful program names)
  - *A README file to explain which-is-which and how your programs can be run*
- 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 program will be tested on the GAUL network. If you will develop your program in any other platform, make sure that your program is error free and produces the required outputs when compiling and running it 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.*

1. (35 marks) Develop the following function:

   void lossless_DPCM_Encoder(char *in_PGM_filename_Ptr,
   int prediction_rule, char *out_filename_Ptr)

   This function should:
   - Read from disk *in_PGM_filename_Ptr image
   - Encode the read image using lossless DPCM encoding, where
     - if prediction_rule = 1, use W to predict encoded pixels
     - if prediction_rule = 2, use N to predict encoded pixels
     - if prediction_rule = 3, use W/2 +N/2 to predict encoded pixels
     - if prediction_rule = 4, use CALIC initial prediction ***as described in the lecture*** (*binary* mode and *continuous-tone* mode using GAP *only*) to predict pixels to be encoded

128	W	W	W	W	W	W	W	W	W
N	N	N	N	N	N	N	N	N	N
N	N								N
N	N								N
N	N								N
N	N								N
N	N								N
N	N								N

   - Regardless of the prediction rule that you will use, you have to consider the following:
     - the pixel at **the first row and the first column (i.e., top left corner)** is always predicted as 128
     - the rest of pixels at **the first row** are always predicted as W
     - pixels at **the second row** are always predicted as N
     - the rest of pixels at **the first two columns** are always predicted as N
     - the rest of pixels at **the last column** are always predicted as N
   - Write the encoded image to *out_filename_Ptr file.
     - The header should include all needed information to reconstruct the image.
   - ***Since you were not asked to do codeword encoding, your program will not achieve compression.***

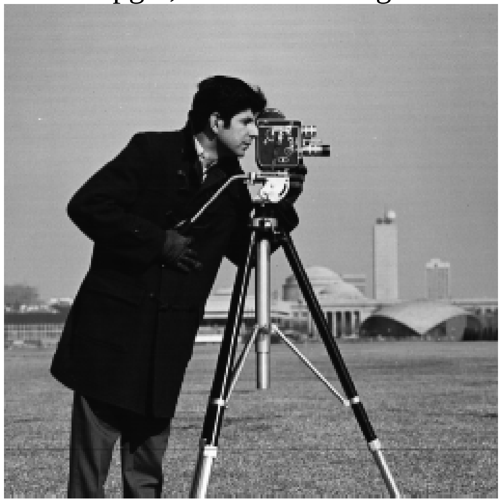
2. (35 marks) Develop the following function:

   void lossless_DPCM_Decoder(char *in_filename_Ptr, char *out_PGM_filename_Ptr)
   This function should:

   - Read from disk *in_filename_Ptr lossless DPCM encoded image
   - Interpret the header that you saved it at the beginning of the file
   - Decode all pixel values
   - Write the reconstructed PGM image to *out_PGM_filename_Ptr file

3. To test your modules, you should use at least the following images:

   - camera.raw.pgm, a 256x256 image: ***You can download a PGM raw version of the image the attachment list.***



   - boats.raw.pgm, a 512x480 image: ***You can download a PGM raw version of the image the attachment list.***

4. (30 marks) ***Document the compressed image structure (<u>header</u> and <u>data</u>) and justify the inclusion and the type of each field in the header.***

- Encode and decode the above images, using
  - prediction_rule number 1
  - prediction_rule number 2
  - prediction_rule number 3
  - prediction_rule number 4
- Make sure that the reconstructed image is EXACTLY the same as the original image.
- Plot *histograms* for the ***<u>ABSOLUTE</u>*** values of the generated prediction errors in *semi-log scale*, i.e., plot the ***<u>ABSOLUTE</u>*** prediction error values against the *log(frequency)*. Note that, ***<u>ABSOLUTE</u>*** prediction error values $\in$ [0, MAX_GRAY_VALUE].

  Generate one histogram per image per prediction rule, i.e., 2 images × 4 prediction rules × 1 histogram per prediction rule per image = 8 histograms in total.
- **Comment** on the shape of the histograms
- Calculate the **mean** and the **standard deviation** of the data that you used to generate each histogram, i.e., calculate 8 histograms × 2 values per histogram = 16 values in total.
- Measure the encoding and decoding time for each image when using various prediction rules, i.e., 2 images × 4 prediction rules per image × 2 times (one for encoding and the other for decoding) = 16 values in total. **You should do so inside your code, not by using a stop watch.**
- Present all these numbers in a **table** that is **easy to read and understand**.
- In your opinion, **which** prediction rule is **more suitable for each image**?
  Justify your recommendation.

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 acceptance of the input from the command line

- README file
- The required compressed/decompressed images
- The required comparison between decompressed image and the original image

- The neatness of caption of your figures
- The neatness of your report

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