Question 2: Steganography using 4-bit LSB algorithm

Scope: File Processing, Pointers, Dynamic Memory Allocation (No arrays)

File to use: stego_lsb.c

Preliminary:

This question should be solved using dynamic memory allocation along with pointers. Any submitted work which includes an array **will be given automatically a zero mark.** To view pgm files, you can use the "Irfan View" software (https://www.irfanview.com/). To read the content of ASCII PGM file, open the file using Notepad or a similar text editor.

The format of ASCII (text format) PGM image file is:

P2

 # comment

Width Height

Maximum Gray value

Pixel values

The format of a Binary PGM image file is:

P5

 # comment

Width Height

Maximum Gray value

Pixel values

Task Description:

Steganography is the art and discipline of embedding hidden messages (**secret message**) into signals (**cover signal**) in such a way that no one suspects the existence of the message and only the sender and recipient can extract it.

In 8-bit images (256 Gray levels), each image pixel value is represented in 8 bits. Humans can't notice a minor difference in the values of the image pixels. Research has shown that altering up to the four Least Significant Bits (LSBs) of the original image pixel values does not significantly distort the original image.

Based on this principle, the **steganography LSB algorithm** when applied on images generates a **stego image** whereby the binary representation of every pixel value has two fields:

1. The upper field includes a number (N) of the **most significant bits** of the **corresponding pixel** in the **cover image**
2. The lower field includes "8-N" of the **most significant bits** of the **corresponding pixel** in the **secret image**

Figure 1 gives an example of the LSB steganography **encoding** with N=4.

Cover Image Pixel: 10110001

Secret Image Pixel: 00111111

Stego Image Pixel: **10110011**

Figure 1. Steganography encoding using LSB algorithm with order 4

When this procedure is completed, an entire image (the secret image) can be hidden within another (cover image) without much noticeable deterioration of either image; allowing a secret message to be passed.

At the recipient, each 8-bit **stego image** pixel is processed as shown in Figure .2 to extract the corresponding pixel value in the **cover image** and the **secret image.**

Each pixel of the **stego image** is split into two 4-bit nibbles, where:

- The most significant 4-bit nibble corresponds to the most significant 4-bit nibble of the decoded **Cover image**
- The least significant 4-bit nibble corresponds to the most significant 4-bit nibble of the extracted **Secret image.**

The least significant 4-bit nibbles in both the Secret and Cover images are set to zero.

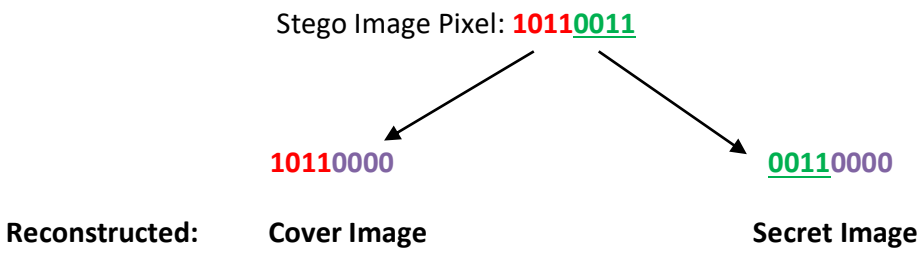

Figure 2. Steganography decoding using LSB algorithm with order 4: the 4 LSBs are all set to zero

Requirements

Complete the attached program: `stego_lsb.c` to implement the 4-bit LSB Steganography encoding and decoding algorithms in C as explained above. The cover image file is "baboon.pgm", while the secret image file is "farm.jpg", both are in ASCII format. **The structure of the attached program should not be altered. Comments in the provided program indicate where additional code needs to be written, as broadly explained below. Further requirements and constraints are detailed bellow:**

- The provided images and those to be generated are all 512x512 pixels in size.
- Using **malloc** function, dynamic memory space should be requested to store the pixels of the cover, secret, stego, and extracted secret images.
- The dynamically allocated memory should be accessed and manipulated using **pointer with offset addressing, not arrays:**
    - The pointer 'coverPixels' should point to the cover image memory space, then to the 'stego' image's.
    - The pointer 'secretPixels' should point to the secret image memory space.
    - The pointer 'outputPixels' should point to the extracted secret image memory space.
- The 'stego' image should be stored in binary format to the file 'stego_image_bin.pgm'
- The extracted 'secret' image should be stored in text format to the file 'extracted_secret.pgm'
- The function 'embedLSB' should be implemented and called to generate the 'stego' image.

- The function ‘extractLSB’ should be implemented and called to extract the ‘secret’ image from the ‘stego’ image.
- The functions ‘readPGMText’, ‘writePGMText’, and ‘writePGMBinary’ should be completed and used to read the provided ASCII images and generate the binary format ‘stego’ image and the text format ‘secret’ image.
- Your code should handle all possible errors that can occur while processing files and pointers, printing relevant messages and returning different values from each function in each case.
- For file processing, include C statements to verify if the corresponding operations were successful.

Hint: you can use C bitwise and shift operators to implement the LSB steganography algorithms.

What to submit

- A description (flowchart or pseudo code) of your design solution to,
  - generate the ‘stego’ image pixel value from the ‘cover’ and ‘secret’ images
  - Extract the ‘secret’ image pixel from the ‘stego’ image
- Full code concisely commented.
- The original ‘cover’ and ‘secret’ image files, along with the ‘stego’ image and ‘extracted secret’ image files.