CSC112 Fundamentals of Computer Science Spring 2016
Lab 6 – 2D Arrays: Hiding Messages in Pictures (Steganography)

You may have heard of a technique for hiding messages in digital photographs. The basic idea is that the human eye is not sensitive to small changes in the brightness level of individual pixels in a digital image. Therefore, the low-order bits of a pixel value may be used to store information.

For simplicity, we will consider only gray-scale images (a.k.a. "black and white" photos) in this lab. A digital image is represented as an $m \times n$ 2D-array of integers. Each integer is in the range 0 to 255, where 0 represents black and 255 is the brightest white. Values from 1 to 254 represent 254 shades of gray. We will hide a message by putting successive bits from the secret message into the lowest-order bit of each pixel value. Let's agree that all messages are a maximum of 140 characters[1] in length. Messages end with the NULL character.

Messages size 140 (plus the NULL character) will require a minimum $8 \times 141 = 1128$ pixels in the image since there are 8 bits in each character, and we are using one bit per pixel. If the image is too small for the message, print an error message and exit.

To simplify input, each image will be stored in a file. The first two numbers in the file are the resolution: i.e.: m rows and n columns. The remaining mn numbers are the pixel values in row-major order.

Your Task; Write two C++ programs: one named `hide.cc` and one named `reveal.cc`. You can name the programs using the `-o` compiler option. For example:

```
gottlieb% g++ -o hide hide.cc
```

Requirements for Program `hide`:

1. Your program must take information from the command line. Command line parameters appear in the following order:

   - input file name
   - output file name
   - message

   For example:

   ```
   hide picture0 picture0.out "Hello, it's me."
   ```

   Using `argc` and `argv` to access words on the command line will be discussed in class.

2. Your program must read (and dynamically allocate) a 2D matrix from the input file.

3. If the file is missing, your program must print an error message and exit.

4. Your program must check that the image is large enough to hide the message. If not, print an error message and exit.

[1] Long enough for Twitter → long enough for us.

5. Your program must hide each bit of the message in the low order bit of each successive pixel in **column major order**. I.e., successive bits are stored in successive rows scanning down each column.

6. Your program must write the image with the hidden message in a indicated on the command line. Your program must store the output file in the same format as the input file (i.e., in row-major order).

7. Remember to hide the NULL character in the picture at the end of the message. The encoded NULL character is needed by program `reveal` to recognize the end of message.

8. Your program must be written in object oriented style using at least one C++ class.

Requirements for Program `reveal`:

1. Your program must take information from the command line. There is only one parameter for program `reveal`, the input file name. For example:

```
reveal picture1
```

2. Your program must extract the secret message and print it to the screen (via `cout`).

3. Your program must read (and dynamically allocate) a 2D matrix from the indicated file. If the file is missing, your program must print an error message and exit.

4. Your program must scan each column and extract the characters hidden in the low order bits of each pixel. You should store each character in an array. Stop when you find the NULL character, or 140 characters, which ever comes first.

5. Some pictures will not contain messages. Your program must scan the extracted message and check for non-printable characters (use function `isprint`). Remember `#include <cctype>` . If the extracted characters contain any non-printable characters, your program should print `No message.`.

6. If the extracted characters are all printable (according to function `isprint`), then print the message.

7. Your program must be written in object oriented style using at least one C++ class.

Suggestions for Your Implementation:

Create a class `image`. Class `image` includes

1. Private data members m and n for the size of the image.

2. Private data member img which is a dynamically allocated 2D array of integers.

3. Public function to read (and allocate) an image file:

```
void read_image(char * filename) ;
```

4. Public function to write an image file:

```
void write_image(char * filename) ;
```

5. Public function to hide a message:

```
void hide_message(char * msg) ;
```

6. Public function to extract a message:

```
void reveal_message(char * msg) ;
```

7. Other public or private data and functions to complete the design of your class.

Testing your Programs:

1. Download a file named "picture0" from menehune.opt.wfu.edu. Run your program using the supplied picture, the output file name "picture0.out" and any message you prefer. It's now time to look at the pictures to see if there is any visible difference. There is a custom program named lab6ppm to convert our simple text-based representation into a gray-scale PPM format image. Use the `display` command to visualize the images. For example:

```
gottlieb %  lab6ppm picture0
gottlieb %  display picture0.ppm &
gottlieb %  lab6ppm picture0.out
gottlieb %  display picture0_out.ppm &
```

2. Download three files named "picture1", "picture2", and "picture3" from menehune.opt.wfu.edu. Run your program `reveal` on each of the files. Make note of the secret message (if any). *Hint: Two have messages, one does not.*

Turn In;

Question 1: Is there any visible difference between the two pictures, with and without the hidden message ?

Question 2: For each of the sample pictures named picture1.out, picture2.out, and picture3.out, what is the hidden message ?

Save all your work in a directory named "Lab6". Change to your home directory (the parent directory of "Lab6"), and create a file named "lab6.tar" using the command:

```
tar cf lab6.tar Lab6
```

Use `sftp` to upload the file "lab6.tar" to your account on telesto.

Spooky Secret Wall Guy