F28HS2 Hardware Software Interface
C Course Work

This course work involves writing a simple steganography system in C.

Steganography is a technique for hiding text in images, by replacing successive random pixels in the image with letters from the text. The text is then retrieved by comparing the new image with the old image, and extracting letters from the new one where it differs from the old one.

You will work with PPM format images. PPM is a very simple open source RGB colour bitmap format. PPM files have contents:

`P3`
`#` *comment*$_1$
`...`
`#` *comment*$_N$
width height
max
r_1 g_1 b_1
r_2 g_2 b_2
r_3 g_3 b_3
`...`

where:

- `P3` – 2 letter code for PPM format
- *width* – integer number of columns
- *height* – integer number of rows
- *max* – integer maximum colour value – usually 255
- r_i g_i b_i – integers between 0 and *max* for pixel *i*'s red, green and blue values

A) design a `struct PPM` to hold PPM images;

B) write functions:

- `struct PPM * getPPM(FILE * fd)`

  to return a PPM image from file `fd;`

- `showPPM(struct PPM * i)`

  to display the PPM image `i` as text in the above format;

- `struct PPM * encode(char * text, struct PPM * i)` to return a copy of PPM image `i` with message `text` hidden in the red field;

- `char * decode(struct ppm * i1, struct ppm * i2)`

  to return the message hidden in the red field of PPM image `i2` by comparison with the red field of PPM image `i1`;

C) write a program `steg` to encode and decode messages in the red field of PPM images.

For the call:

```
$ steg e file1.ppm > file2.ppm
```

`steg` prompts for a message and shows *file*$_1$`.ppm` with the message encoded within it. Here, the output is redirected to *file*$_2$`.ppm`.

For the call:

```
$ steg d file1.ppm file2.ppm
```

`steg` shows the message hidden in *file*$_2$`.ppm` after decoding by comparison with *file*$_1$`.ppm`.

You should submit your:

- program design, outlining your choice of data structures and algorithms;
- program listing;

by Friday 3rd March.

Subsequently, you will be required to demonstrate your program to Greg (Edinburgh) or Hani (Dubai) , on unknown PPM files which they will provide.

Notes:

1. There's a description of PPM at http://netpbm.sourceforge.net/doc/ppm.html
2. You can view PPM files with `gimp` on Linux or ImageViewer on Raspbian.
3. You can use `od` to look at what's inside a PPM file. For example:

```
$ od -x -a ape.ppm | more
```

4. The C random number library is in `stdlib`. Calling `srand(int)` will seed the random number generator with the `int`. Subsequent calls to `rand()` will return a number in the range `0` to `RAND_MAX`. So you can get a value in the range `1` to `N` from `rand()%N+1`.
5. Your PPM representation should keep all the comments.
6. Your program should be able to deal with PPM files with arbitrary numbers of rows and columns.