Lab 8: Grover's Grove Maps...

Due date: Tuesday, November 12, **in class**

Lab Assignment

Assignment Preparation

Lab type. This is a **pair programming lab**. For this lab, select a partner who was not your partner for Lab 7 and who is not your partner for program 1.

Collaboration. Students work in pairs, and it is considered cheating, if members of the team (pair) do not work together. Communication between pairs during lab time is allowed, but no direct sharing of code is allowed.

Purpose. The lab allows you to practice the use of loops.

Programming Style. All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/~dekhtyar/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a penalty. Significant stylistic violations, epsecially those that make grading harder, may yield reasonably strict penalties.

Testing and Submissions. Any submission that does not compile using the

```
gcc -ansi -Wall -Werror -lm
```

compiler settings will receive an automatic score of 0.

For each program you have to write for this lab, you you are provided with a copy of instructor's output, the PPM file produced by instructor's executable. The programs you submit must output exactly the same PPM file.

Program Outputs must co-incide. Any deviation in the output is subject to penalties.

Note: The difference between PPM files cannot be easily checked using Linux `diff` command, however, it can be observed in a straighforward manner by loading two images into an image browser and viewing them in rapid succession.

Please, make sure you test all your programs prior to submission!

Portable Pixel Maps

PPM Format Explanation

Portable Pixel Map (`.ppm`) file format is a simple format for storing graphical images. Files in this format can easily be created using C programs.

Basics. A computer image is a two-dimensional grid of pixels. Each pixel represents the smallest undivisible part of the computer screen. An image file is an assignment of color to each pixel. Pixels are referred to by their Cartesian coordinates. The top left corner of an image has the coordinates (0,0). The bottom right corner has the coordinates $(n-1, m-1)$ where n is the width of the image in pixels and m is the height of the image in pixels.

Colors. Portable pixel map files use RGB (Red, Green, Blue) color format to represent the color of each pixel. In RGB format, a color of a single pixel is separated into three components: the red component, the green component and the blue component. The final color of an RGB pixel is determined by combining the Red, Green and Blue components into a single color.

In our course, all individual RGB component intensities range from 0 (not visible) to 255 (highest intensity) and are represented as integer numbers. RGB color (0,0,0) is black, RGB color (255,255,255) is white. The table below contains the list of colors used in this lab and their RGB values.

Color	RGB Red	RGB Green	RGB Blue
black	0	0	0
white	255	255	255
red	255	0	0
green	0	128	0
blue	0	0	255
light blue	0	127	255
yellow	255	255	0
purple	255	0	255
orange	255	128	0

File format. There are two PPM formats: a "raw" PPM file and a "plain" (ASCII) PPM file. ASCII PPMs are human-readable, but they take too much space. Raw PPMs are smaller in size, but cannot be read by a human. In this lab you will be generating raw PPM files.

From http://netpbm.sourceforge.net/doc/ppm.html (with some modifications):

Each PPM image consists of the following:

1. *A "magic number" for identifying the file type. A ppm image's magic number is the two characters* `"P6"`.
2. *Whitespace (blanks, TABs, CRs, LFs).*

3. *A width, formatted as ASCII characters in decimal.*
4. *Whitespace.*
5. *A height, again in ASCII decimal.*
6. *Whitespace.*
7. *The maximum color value (Maxval), again in ASCII decimal. Must be less than 65536 and more than zero.*
8. *A single whitespace character (usually a newline).*
9. *A raster of Height rows, in order from top to bottom. Each row consists of Width pixels, in order from left to right. Each pixel is a triplet of red, green and blue intensities, in that order*

PPM File header example. The first three lines of a PPM file containing the magic number, the height, the width and the Maxval values is called the PPM file header. A sample header, for a PPM file describing a 600 by 900 image that consists of standard RGB colors in the range of 0 to 255 is shown below:

```
P6
600 900
255
```

Representing Colors in C. Outputing raw PPM files is actually quite simple. The idea is to use `unsigned char` variables to store information about RGB intensities.

Variables of type `char` and `unsigned char` are treated by C both as a character and as a number in the range -128 – 127 or 0 — 255 respectively. The following code outputs an RGB triple to stdout.

```
unsigned char Rcolor, Bcolor, Gcolor;
Rcolor = 255;
Bcolor = 0;
Gcolor = 128;

printf("%c%c%c", Gcolor, Bcolor, Rcolor);
```

In addition to this, you can use C's `#define` directives to define color "substitutes" as follows:

```
#define RED 255,0,0
```

Constants defined this way, CANNOT be used in assignment statements, i.e., `unsigned char color = RED;` is an INCORRECT assignment. But, they can be used in `printf()` statements as follows:

```
printf("%c%c%c",RED);
```

(note, that after the preprocessor has finished its work, this statement turns into `printf("%c%c%c",255,0,0)`, which is exactly what is desired.

Packing pixels into a PPM file. The information in the PPM file header determines how many pixels are in the image. For example, the header above specifies that there are 600 columns and 900 rows in the image for a total of $600 \cdot 900 = 540000$ pixels. The PPM files you are building represent each pixel color as three values as discussed above, so, a total of $540000 \cdot 3 = 1,620,000$ color codes need to be placed in the body of the PPM file.

As stated in the format description, the order in which pixel colors are defined is preset:

- The first triple of color intensities describes the color of the top left corner of the image (row 1, column 1).
- The second triple describes the color of the next pixel in the same row (row 1, column 2).
- ...
- The triple number *width* describes the color of the last pixel of the first row, while the triple number $width+1$ describes the color of the first pixel of the second row (row 2, column 1).
- ...
- The last triple of color intensities describes the color of the bottom right corner of the image (row *height*, column *width*).

Note: PPM files can contain ONLY the information specified above. E.g., no `line breaks` are allowed in the body of the PPM file (you can and will put linebreaks in the headers, but once you start outputting pixel colors, every single thing you print out will be treated as a pixel color).

Creating PPM images via output redirection. C programs can generate PPM images by printing the contents of the image file, *starting with the PPM image header* to `stdout`, i.e., the standard output. The actual PPM file is then created by redirecting the output to a file with a specified file name.

E.g., if a C program `image.c`, compiled into an executable `myImage`, prints out the contents of a specific image using `printf()` statements, this program can be run to produce a PPM image file `myImage.ppm` as follows:

```
> myImage > myImage.ppm
```

Viewing PPM files

On our Linux system, PPM files can be viewed using the default picture viewer (e.g., when double-clicking the PPM file icon in the directory explorer). The default picture viewer is `eog`, a.k.a., Eye of Gnome. To use it from command line, type:

```
> eog <fileName>.ppm &
```

On Windows, standard Windows picture viewers do not recognize PPM format. However, freeware picture viewers exist. One such viewer, `XnView` can be downloaded from

http://www.xnview.com/en/xnview.html

Note, XnView is also available for MacOs, although I have no experience running it on Macintoshes.

General Instructions

1. All programs that output ppm images should use stdout. We will be creating files using output redirection.
2. For this assignment, use exactly the colors specified in the color table above.
3. All images you are asked to produce are also available from the course web page. Please refer to the images, if the textual descriptions of the flags are insufficient.

Programs

You need to write just two programs for this lab. Each program will combine your mastery of the PPM file format and your Grover's Grove Game knowledge. Each of the two programs outputs a PPM image of one maps of Grower's Grove, in which each field is colored to represent a certain value. For this lab, you will create the following two maps:

1. Soil Quality map. (C program `soilMap.c`).
2. Field Profit map. (C program `profitMap.c`).

Each PPM image you create needs to abide by the following rules:

1. **Image size.** The image is 800×800 pixels.
2. **Field representation.** Each field is represented by a 40×40 block of pixels. The field $(1, 1)$ is in the bottom left corner of the image, $(1, 20)$ is the top left corner, $(20, 1)$ is bottom right corner and $(20, 20)$ is top right corner.
3. **Field image.** Each field image consists of a "background" frame and the foreground color box. The background frame is described as follows:
   - **width:** 3 pixels wide;
   - **color:** light gray: (R,G,B) = (196,196,196).

   The foreground box will always be a solid color. The color is determined by each program according to the value of the appropriate parameter (soil quality, profit) for the current field.

Use of arrays. We can represent a PPM image as a 3-dimensional array

```
#define COLORS 3

char image[HEIGHT][WIDTH][COLORS];
```

where `WIDTH` and `HEIGHT` are constants representing the width and the height of the image in pixels. The third dimension of the array is the number of color components (`COLORS` is set to 3). For each pair of values `x,y`, `image[x][y]` stores the R,G and B components of the color. By convention:

`image[x][y][0]`	R(ed) component
`image[x][y][1]`	G(reen) component
`image[x][y][2]`	B(lue) component

Soil Quality Map

Your soil quality map program (`soilMap.c`) shall assign the color to each field as follows:

- **Overall color pallette.** The soil quality will be encoded in shades of green. Therefore, all colors you assign will have the form (0, X, 0), where X is the intensity of the shade of greed used to illustate the quality.
- Lighter green means better quality, darker green means lower quality of soil.
- The specific shade of green, X is determined using the following process.
  1. Find the best soil quality in the Grower's Grove ($maxQ$).
  2. Set, as the lowest possible soil quality, the value 0 ($minQ$).
  3. Find the soil quality of the current field $(x, y) : q = soilQuality(x, y)$.
  4. The shade X is then computed as

$$\mathsf{X} = \frac{q - minQ}{maxQ - minQ} \cdot 255.$$

Profit Map

Your profit map program (`profitMap.c`) shall assign the color to each field according to the table below.

Profit range	Color name	Red	Green	Blue
$[500, \infty)$	Dark Green	0	128	0
$[300, 500)$	Lighter Green	0	200	0
$[100, 300)$	Light Green	0	255	0
$[10, 100)$	Pale Green	128	255	128
$[0, 10)$	Off-white Green	220	255	220
$[-10, 0)$	Off-white Red	255	220	220
$[-50, -10)$	Pale Red	255	128	128
$(-\infty, -50)$	Red	255	0	0

Testing

Instructor's images are provided. Use them to compare your output.

Suggestion. First, write a program that outputs the frames and the boxes for each field correctly, using just one base color for boxes. Then, add code that computes the appropriate color for each pixel.

Design. You can develop any functions to help you solve the problem. Among the functions you may find helpful are:

- a function to color a rectangle (square) of space in the array a specific color;
- a function that determines a color of the box for a given field;
- functions that determine the X and Y coordinates of the graphical representation of a given field (x, y).

You may also find other functions useful.

Compilation and running. Compile and run your code as follows:

```
$gcc -ansi -Wall -Werror -lm -o soilMap soilMap.c grove.c
$ soilMap > soilMap.ppm
$gcc -ansi -Wall -Werror -lm -o profitMap profitMap.c grove.c
```

Submission

Submit the following files: `grove.c`, `grove.h`, `soilMap.c`, `profitMap.c`.

Ssh to `unix1, unix2, unix3` or `unix4` and submit using the following command:

```
$handin dekhtyar lab08 <files>
```