Lab 9: Ifs and Loops...

Due date: Monday, February, 13, beginning of the lab period.

Lab Assignment

Assignment Preparation

Lab type. This is a **pair programming lab**. For this lab, the partner gets assigned to you randomly.

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/~cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

Note: This lab involves use of a lot of integer (and character) constants in the code. Make sure you `#define` all of them.

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.

Programming notes. All programs below shall be written as a single `int main()` function, with no other functions defined. **Do not** use arrays in this assignment. Our next lab will introduce the use of arrays for image construction, but for this lab, we are sticking with generation of output on the fly.

Image 1: Flag of Russia russia.c

Your first image is a flag of Russia. The flag of Russia consists of three equal-sized horizontal fields. The colors (top-to-bottom) are `white`, `blue`, `red`.

Your program shall output the .ppm file representing the Russia flag. Name your program `russia.c`.

Image dimenstions: 800 (columns) $\times$600 (rows).

Image 2: Flag of Italy: italy.c

The image is the flag of Italy. The flag of Italy consists of three equal-sized vertical fields. The colors (left-to-right) are `green`, `white`, `red`.

Your program shall output the .ppm file representing the flag of Italy. Name your program `italy.c`.

Image dimenstions: 600$\times$400.

Image 3: Flag of The United Arab Emirates: uae.c

The image is the flag of The United Arab Emirates. The flag consists of a vertical red field at the hoist (left side of the flag), and three horizontal fields of equal height at the fly (right side of the flag). The colors of the horizontal fields are from top to bottom: green, white, black.

Feature	Size/Dimensions
Image size:	1200 columns $\times$ 600 rows
Vertical field width:	300 columns

Name your program `uae.c`

Image 4: Flag of Finland: finland.c

The image is the flag of Finland. The flag is a white field with a `blue` cross on it.

Image dimensions:	900×550.
Cross bar width:	150.
Left rectangles (top and bottom):	250×200
Right rectangles (top and bottom):	500×200

Image 5: Flag of Laos: laos.c

The image is the flag of Laos. The flag of Laos consists of three horizontal stripes: `red` at the top and bottom, `blue` in the middle, with a white circle centered inside the blue stripe. The dimensions are as follows:

Image dimensions:	600×400
Red stripe height (top and bottom):	100
Blue stripe height:	200
Circle radius:	80
Circle center:	$(300, 200)$

Image 6: Flag of Greenland: greenland.c

The image is the flag of Greenland. The flag consists of two equal horizontal stripes: `white` at the top and `red` at the bottom, with a circle on top. The center point of the circle is shifted to the left. The top half of the circle is `red` and the bottom half of the circle is `white` making the top and the bottom halves of the flag negative images of each other. The official flag description is:

> *The flag is 12 parts by 18, the white and red stripe are both 6 parts. The centre of the circle is set 7 parts from the hoist along the dividing line between the white and red, the radius being 4 parts. The upper part of the circle is red, the lower white.*

Image dimensions:	600×400
Center of the circle:	200×200
Radius of the circle:	125

Image 7: Flag of the Republic of Congo: congo.c

The image is the flag of the Republic of Congo (capital: Brazzaville). The flag consists of three fields: a green triangle at the hoist, a yellow diagonal bar in the middle, and a red triangle at the fly. Name your program `congo.c`. The dimensions of the flag are:

Image dimensions:	900 columns $\times$ 600 rows
Yellow bar width:	300 columns

(all other dimensions can be deduced from these).

Image 8: Flag of Scotland: scotland.c

The image is the flag of Scotland. The flag, otherwise known as "St. Andrew's flag" is a *saltire*, i.e., a *diagonal cross*. On the flag of Scotland, the `white` saltire splits the `blue` field into four triangles.

Image dimensions	500×300
Horizontal offset (white)	50
Vertical offset (white)	30

Note: The horizontal and vertical offsets specify respectively the number of `white` pixels in the first row and the first column from the (1,1) point and to the first `blue` pixel.

Submission.

Files to submit. You shall submit **nine** files:

```
team.txt,
russia.c,
italy.c,
uae.c,
finland.c,
laos.c,
greenland.c,
congo.c,
scotland.c
```

`team.txt` file shall contain the name of the team and the names of the two team members in each pair, and the Cal Poly IDs of each. E.g, if I were on the team with Dr. John Bellardo, my `team.txt` file would be

```
Go, Poly!
John Bellardo, bellardo
Alex Dekhtyar, dekhtyar
```

Files can be submitted one-by-one, or all-at-once, but all files should be submitted from the same student account.

Submission procedure. You will be using `handin` program to submit your work. The submission command is:

```
> handin dekhtyar lab09  <your files go here>
```

Grading

Each program is worth 12.5% of the lab grade.

Any submitted program that does not compile earns 0 points.

All programs will be checked for style conformance. Any style violation will be noted. The program will receceive a 10% penalty. (In particular, declare all "magic" numbers as constants).

Appendix A. Testing

Instructor's Images. The PPM files generated by the instructor's programs are provided to you on the Lab 4 web page.