Lab 4: Ifs and Loops...

Due date: Thursday, October 16, 11:59pm.

Lab Assignment

Assignment Preparation

Lab type. This is an **pair programming lab**. For this lab, you get to select your own partner. The only rule is that your partner **must be different** than your Lab 3 partner. The lab commences immediately after the lab exam, feel free to establish pairs before the lab period starts, or immediately upon completion of the lab exam. See the pair programming handout from the first day of classes for more information on what is expected of you.

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 conditional statements and loops. It also facilitates learning the use of boolean expressions as conditions in `if`, `switch` and `while` statements.

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.

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, you will be provided with instructor's executable and with a battery of tests. The programs you submit must pass all tests made available to you. You can check whether or not a program produces correct output by running the instructor's executable on the test case, then running yours, and comparing the outputs.

Program Outputs must co-incide. Any deviation in the output is subject to penalties. **PLEASE, USE BINARY EXECUTABLES PROVIDED BY THE INSTRUCTOR!**. The exception is made in case of floating point computations leading to differences in the last few decimal digits.

Please, make sure you test all your programs prior to submission! Feel free to test your programs on test cases you have invented on your own[1].

The Task

Note: Please consult the instructor if any of the tasks are unclear.

In this lab you, working in pairs, will perform the following tasks:

1. Write a few more programs recognizing colors for the if-statement zones.
2. Write a few programs that output PPM files that match specific if-statement zone problems.

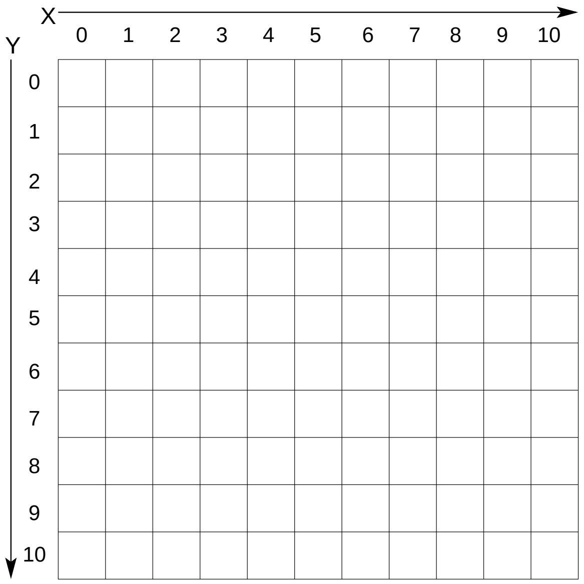
Task 1: More If-Statement Zones

Remember:

If-Statement Zones are regions of a 11×11 two-dimensional grid. The final task of the lab asks you to **recognize** specific if-statement zones from the examples provided to you.

The overall grid, on which the if-statement zones will be marked is shown below:

[1] In this lab, we provide you with a large collection of test cases. In some future labs, test case development will be a major part of the lab assignment, so it never hurts to start early



Each zone on the grid has a `color`. The color of the zone comes from the following list (note, all color names are lowercase):

```
red
green
blue
black
yellow
purple
orange
```

Your task is, given a grid with a number of zones write a program which determines color of each point on the grid.

General Requirements

Z1. If a region has no color, it is assumed blank, your program shall provide no output. There is no "white" color.

Z2. For each program you are given the exact specification of **how** the decisions about the colors need to be made. In particular, we specify the number of `if` and/or `switch` statements, the number of `&&` and the number of `||` logical operators you can use in the program. (Note, in this lab, you are only allowed the use of `if` statements, no `&&` or `||` operators). You may use as many comparison (==, !=) and logical negation (!) operators as well as as many parenthesis as you like.

Z3. Never use an empty `if` or `else` branch in your code ("empty" means a branch with no statements in it).

Z4. All programs shall work in exactly the same manner. The x and y coordinates on the grid are `integers`. The program shall prompt for the x coordinate first (the prompt text is `Enter X:`), read the x value, then prompt for the y

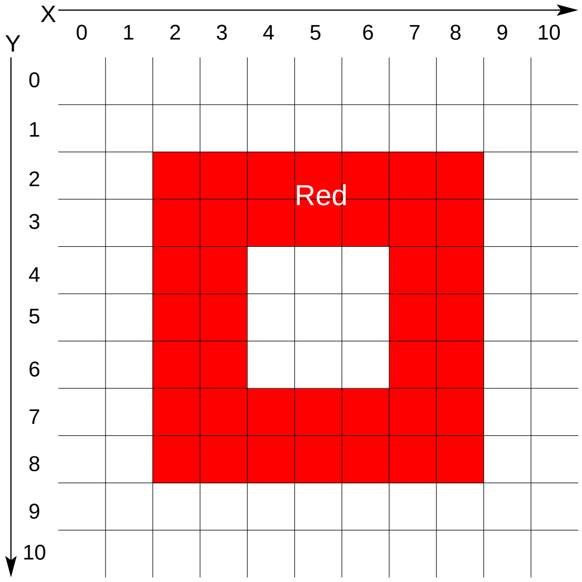
coordinate (the prompt texts is `Enter Y:`), read the y value. After that, each program needs to perform a series of checks to establish the color of the given point. Once the color is determined, it is to be printed out, after which the program should stop its operation.

Z5. All programs shall be **compatible** with `pix2ppm`. This means, **first and foremost** that your program provides exactly the same output as the instructor's binaries.

Program 1: zones11.c

Restrictions:

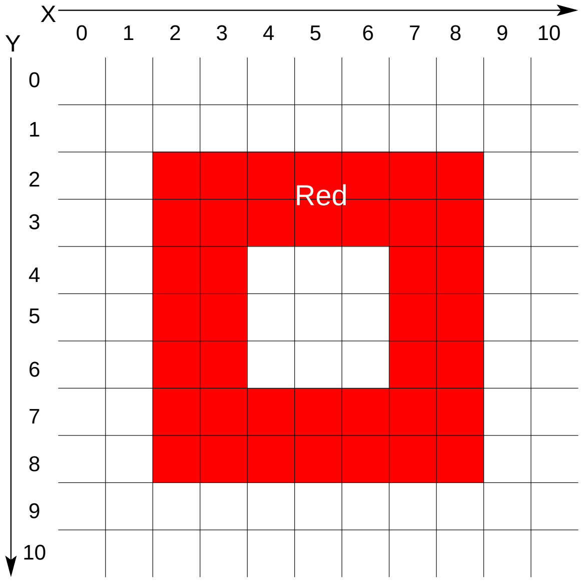
Number of `if`s:	1
Number of `&&` operators:	any
Number of `||` operators:	any



Program 2: zones12.c

Restrictions:

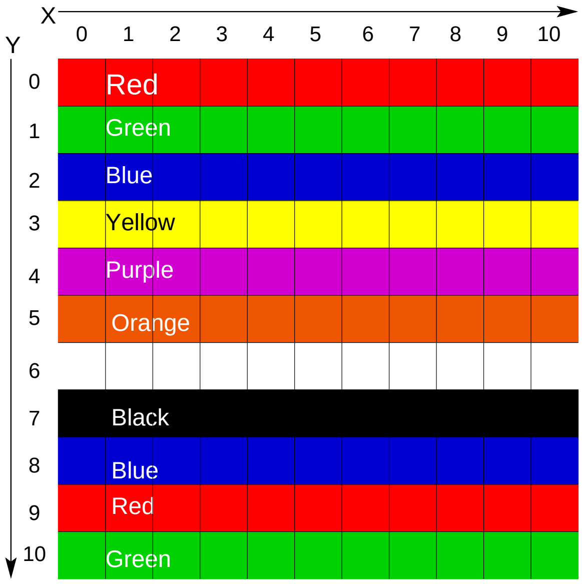
Number of `if`s:	1
Number of `&&` operators:	any
Number of `||` operators:	0



Program 3: zones13.c

Restrictions:

Number of `ifs`:	0
Number of `switches`	1
Number of `&&` operators:	0
Number of `||` operators:	0



Program 4: zones14.c

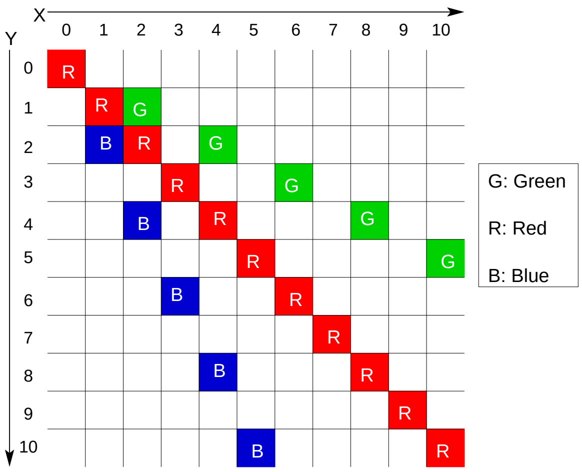
Restrictions:

Number of `ifs`:	as needed
Number of `switches`:	1
Number of `&&` operators:	0
Number of `||` operators:	0

Program 5: zones15.c

Restrictions:

Number of `ifs`:	3
Number of `&&` operators:	0
Number of `||` operators:	0



Task 2: 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. An 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, m) 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	255	0
blue	0	0	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 green, blue and red intensities, in that order*[2]*.*

[2] This is what worked for me. If your colors don't look right, switch to RGB order.

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);
```

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.
4. On images for if-statement zones problems, lack of color for a pixel is represented as coloring it `white`.

Image 1: Russian Flag: 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: 400×300.

Image 2: French Flag: france.c

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

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

Image dimenstions: 300×200.

Image 3: Zone 1: zone01ppm.c

Create a ppm image of the first if-statement zone problem from Lab 2 (program `zones01.c`).

The image dimensions for all if-statement zone problems shall be 440× 440. Each square from the if-statement-zone diagram shall be represented on the .ppm image a 20×20 raster image.

Note, that your image won't contain gridlines. This is the correct expected behavior for all if-statement zones drawing programs.

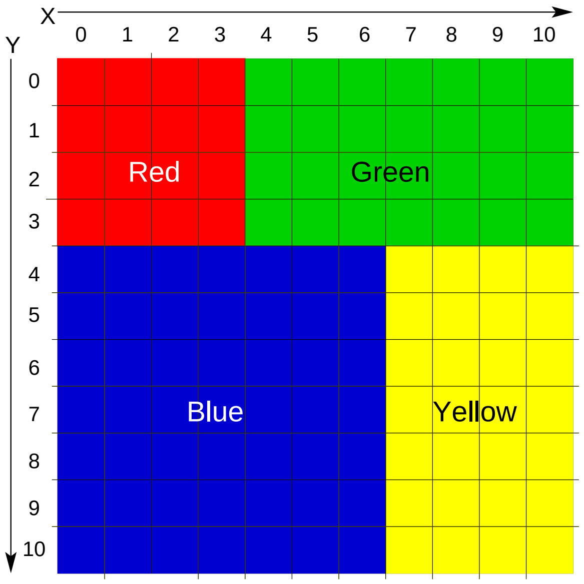


Image 4: Zone 4: zone04ppm.c

Create a ppm image of the fourth if-statement zone problem from Lab 2 (program `zones04.c`).

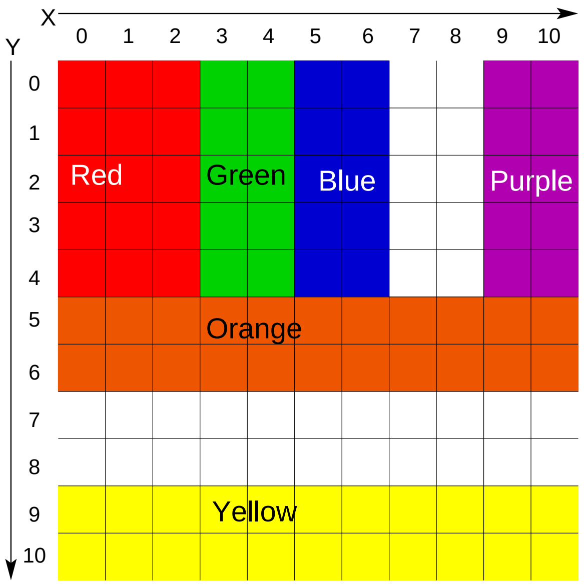
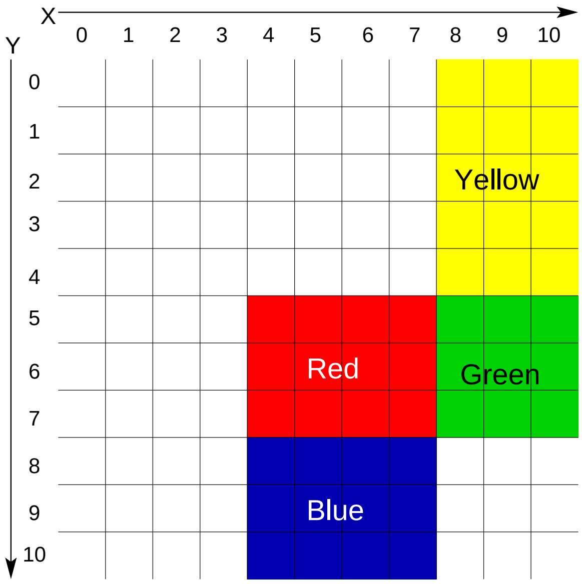


Image 5: Zone 8: zone08ppm.c

Create a ppm image of the third if-statement zone problem from Homework 1 (program `zones08.c`).



Submission.

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

> `team.txt`, `zones11.c`,`zones12.c`, `zones13.c`, `zones14.c`, `zones15.c`, `russia.c`,`france.c`, `zones01ppm.c`, `zones04ppm.c`, `zones08ppm.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
```

No other files can be submitted. In fact, if you submit *other* file, or submit one of the three files about with an *incorrect filename*, you will receive an email informing you about a submission error, and asking you to resubmit.

Files can be submitted one-by-one, or all-at-once.

Submission procedure. Please note the changes!!!

You will be using `handin` program to submit your work. The procedure is as follows:

- `ssh` to `vogon` (`vogon.csc.calpoly.edu`).
- Students from **Section 009** shall execute the following submission command:

  ```
  > handin dekhtyar-grader lab04-09  <your files go here>
  ```

- Students from **Section 011** shall execute the command:

  ```
  > handin dekhtyar-grader lab04-11 <your files go here>
  ```

`handin` is set to stop accepting submissions 24 hours after the due time.

Grading

Each program is worth 10% of the lab grade.

Any submitted program that does not compile earns 0 points.

Any submitted program that compiles but fails at least one **public** (i.e., made available to you) tests earns no more than 30% of its full score (and can possibly earn less).

Any submitted program that compiles and succeeds on **all** publically available tests earns at least 50% of its full score.

The output of your PPM programs will be checked both using `diff` (a program that compares two input files and shows where they are different...).

All programs will be checked for style conformance. Any style violation will be noted. The program will receieve a 10% penalty.

Appendix A. Testing

Instructor's executables. Instructor's executables for all programs are provided to you on the course web page. When copying them, please make sure to run `chmod u+x` on them.

Public test suites. Public test suite for `zones11---zones15` programs is the same as the public test suite for `zonesXX` programs. Additionally, `pix2ppm` will be used to validate your output. For `zonesXXppm` programs, we check the output file against a ppm file created by the instructor's code.