CS 130, Final

Solutions

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Read the entire exam before beginning. **Manage your time carefully.** This exam has 90 points; you need 75 to get full credit. Additional points are extra credit. 90 points → 2 min/point. 75 points → 2.4 min/point.

Problem 1 (3 points)

Most color printers operate in CMYK space (C=cyan, M=magenta, Y=yellow, K=black) and have separate ink/toner cartridges for each of those four colors.
(a) Why are cyan, magenta, and yellow used instead of red, green, and blue?
(b) Why do these devices use a fourth color?

(a) RGB are used for additive color mixing, as occurs with lights. These colors primarily stimulate individual cones in the eye. CMY are used for subtractive color mixing, as occurs with pigments (and printing). These *absorb* individual colors.
(b) Black is used as a separate cartridge because most printed content is in black. Note that black could be obtained by mixing the other colors.

Problem 2 (3 points)

What is an area light, and how can this effect be achieved by a raytracer?

Area lights are lights that occupy a finite area (as opposed to a point light). They are ray traced by casting rays to randomly sampled locations on the light to determine visibility. If a subset of rays see the light, the light will be partially shadowed.

Problem 3 (3 points)

The Lambertian shading model depends on direction to the light but not the viewing direction. Why? (The specular model, by contrast, depends on both.)

The Lambertian model assumes that incoming light is emitted uniformly in all directions, so the viewing angle does not matter. The angle of incoming light determines how much light actually hits an area of surface; if the surface is at a grazing angle the light that comes it will be spread over a larger area of the object.

Problem 4 (3 points)

For each of the following four types of lights we discussed in this course, indicate whether the light falls off with distance (Y or N) and give a brief (at most one sentence) explanation for why: (a) ambient, (b) directional, (c) point, and (d) spotlight.

(a) N. Ambient light is light that is assumed to be coming from all around; moving farther from light from one source moves you closer to light from another.
(b) N. Directional lights are assumed to be so far away that the direction is essentially constant (e.g., the sun); moving a little closer or farther will make no difference.
(c) and (d) Y. This light all comes from one point and spreads out to illuminate a larger area as the distance to the source increases; this spreads out the light's energy and makes its brightness fall off with distance.

Problem 5 (3 points)

For each of the following examples, indicate whether the light source is best described as A=ambient, D=directional, P=point, or S=spotlight. For each, fill your answer (A, D, P, or S) in the table. No explanation is required. Your solution must be in the table to count.

#	object	answer
(a)	flashlight	S
(b)	candle	P
(c)	twilight†	A
(d)	light bulb	P
(e)	moonlight	D
(f)	sunlight	D

†Twilight is the period of the day after sunset (or before sunrise) when the sun is not visible in the sky but it is also not dark.

Problem 6 (3 points)

During which pipeline stage would a normal map be implemented? Explain your answer.

This would be implemented in the fragment shader (as regular texture mapping is) since it must be performed per pixel and those computed normals would feed into the user-defined shading.

Problem 7 (3 points)

Why do we need to clip before the perspective divide?

Some critical information is lost during the perspective divide. For example, the segment connecting $(2, 0, 0, 1)$ and $(2, 0, 0, -1)$ should be completely discarded (it fails $x < w$). After the perspective divide, these become $(2, 0, 0)$ and $(-2, 0, 0)$, which should be clipped to $(1, 0, 0)$ and $(-1, 0, 0)$, resulting in a segment across the middle of the image.

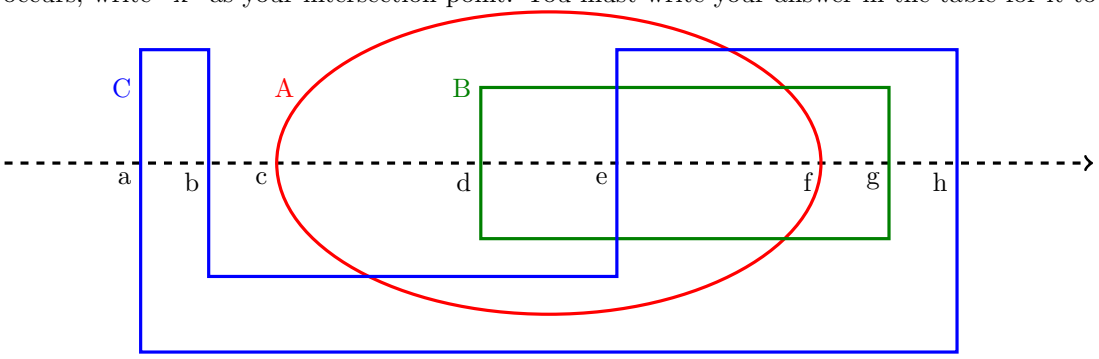
Problem 8 (3 points)

The graphics pipeline contains many stages, including the following: (a) perspective divide (divide by w), (b) geometry shader, (c) vertex shader, (d) rasterization, (e) z-buffering, (f) fragment shader, (g) clipping, and (h) tessellation shader. List these stages in the order that they are performed in the pipeline. If two steps may be performed in either order, you may select an order arbitrarily. No explanation is required.

Order: c, h, b, g, a, d, f, e. Under some circumstances, f and e can be reversed as an optimization.

Problem 9 (5 points)

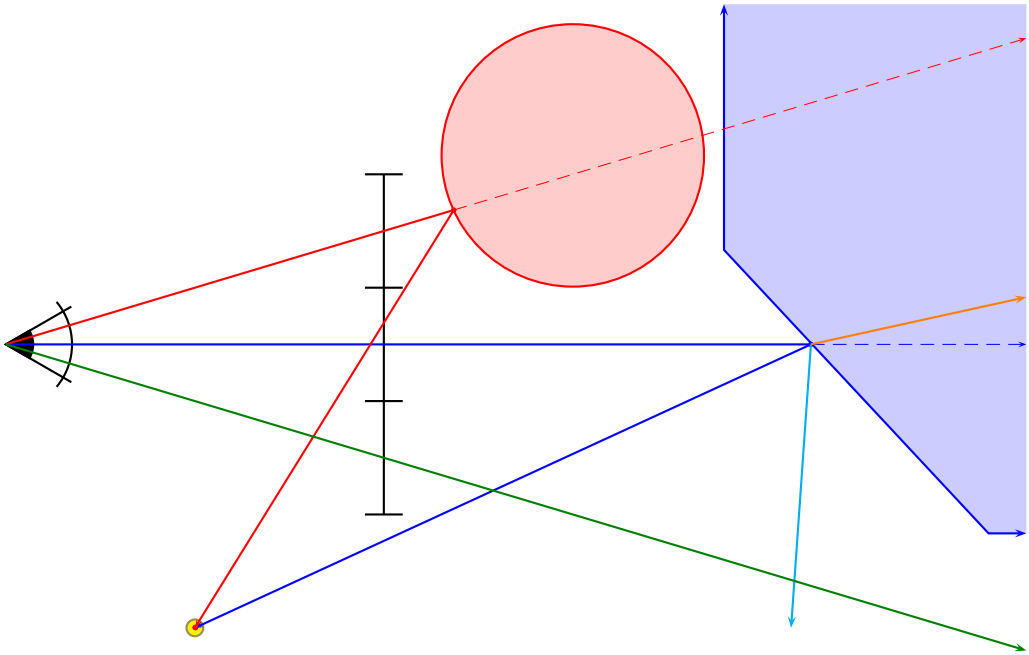
You are given the ray below (dashed) and objects A, B, and C. The intersections between the individual objects and the ray are labeled (a-h). For each of the Booleans composite objects below, identify the closest intersection point in the table. It is sufficient to write its label; no explanation is required. If no intersection occurs, write "x" as your intersection point. You must write your answer in the table for it to count.



#	object	answer
-	$A \cup B \cup C$	a
(a)	$A \cap B \cap C$	e
(b)	$A - (B \cup C)$	c
(c)	$B - (A \cup C)$	x
(d)	$B - (C - A)$	d
(e)	$B - A$	f

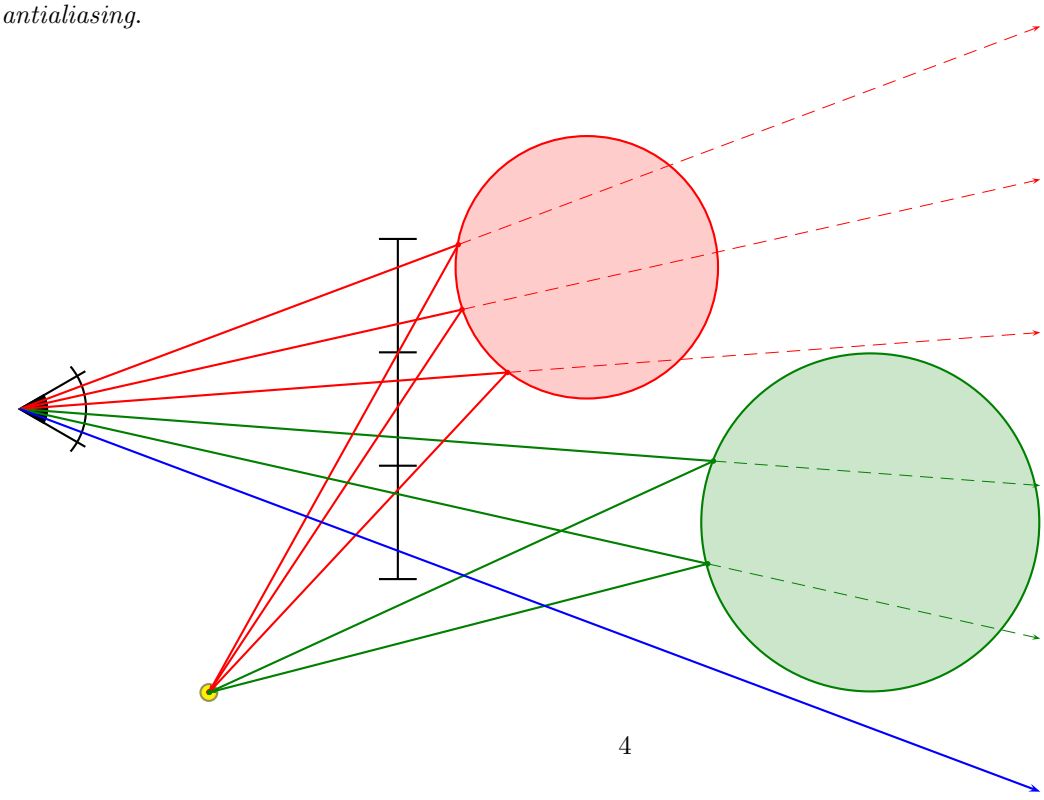
Problem 10 (6 points)

Below is a simple 2D raytracing setup. The 1D image has three pixels. The blue object is made of glass (reflective and transparent). The red object is made of wood. The yellow circle is a point light. Draw all of the rays that would be cast while raytracing this scene.



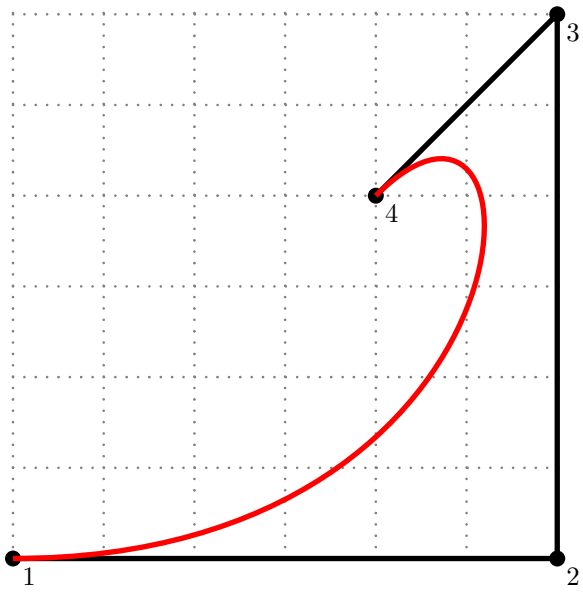
Problem 11 (6 points)

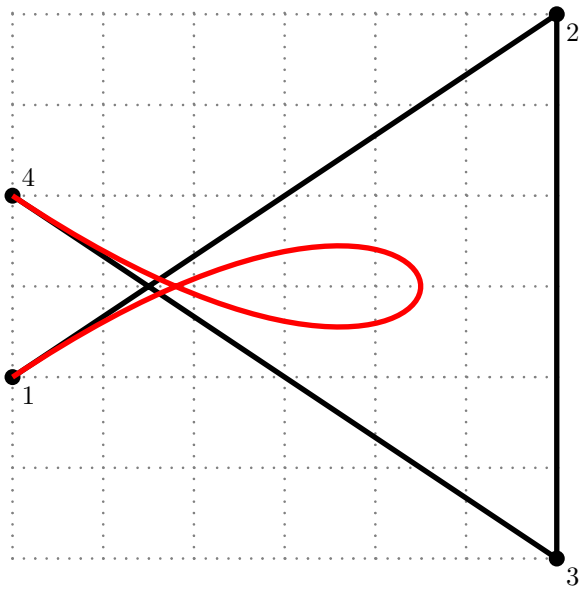
Below is a simple 2D raytracing setup. The 1D image has three pixels. Both objects are made of wood. The yellow circle is a point light. Draw all of the rays that would be cast while raytracing this scene *with antialiasing*.

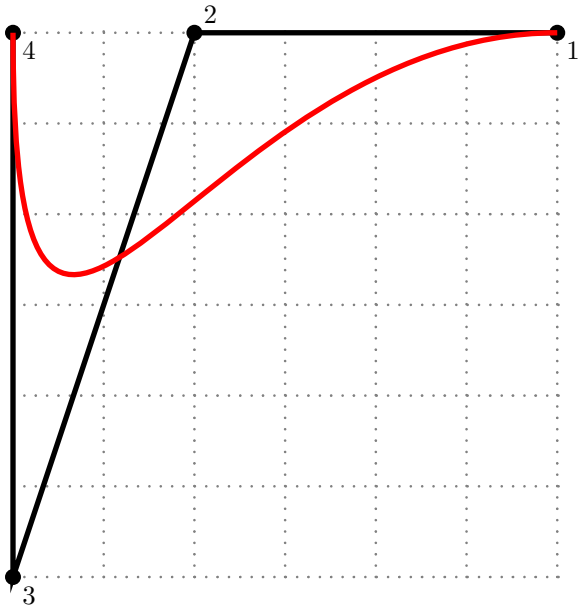


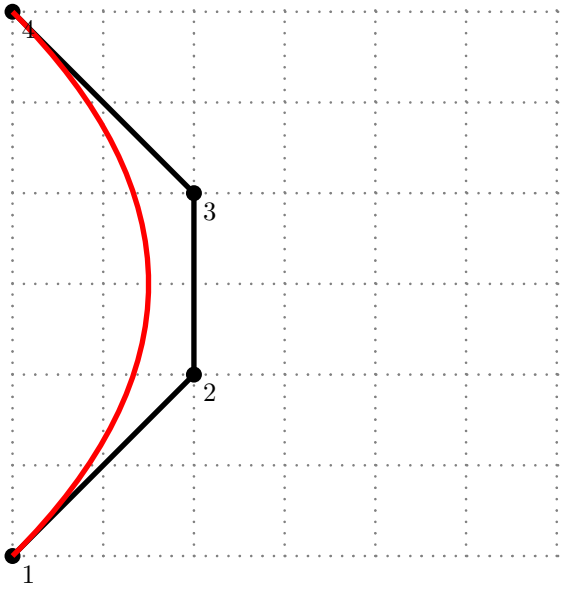
Problem 12 (6 points)

In each of the four examples below, control points are shown for a cubic Bezier curve. Sketch out approximately what these curves will look like.



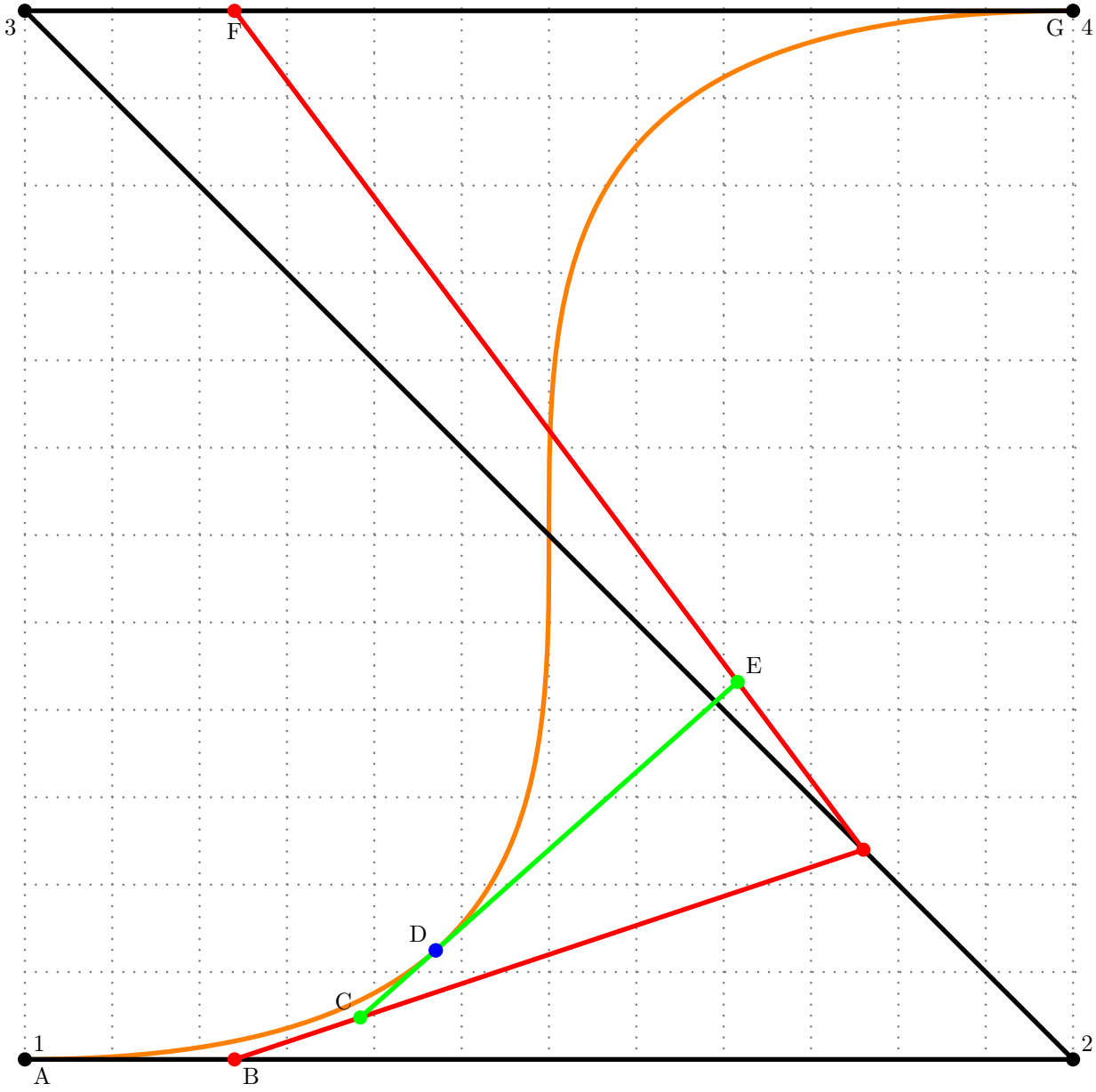






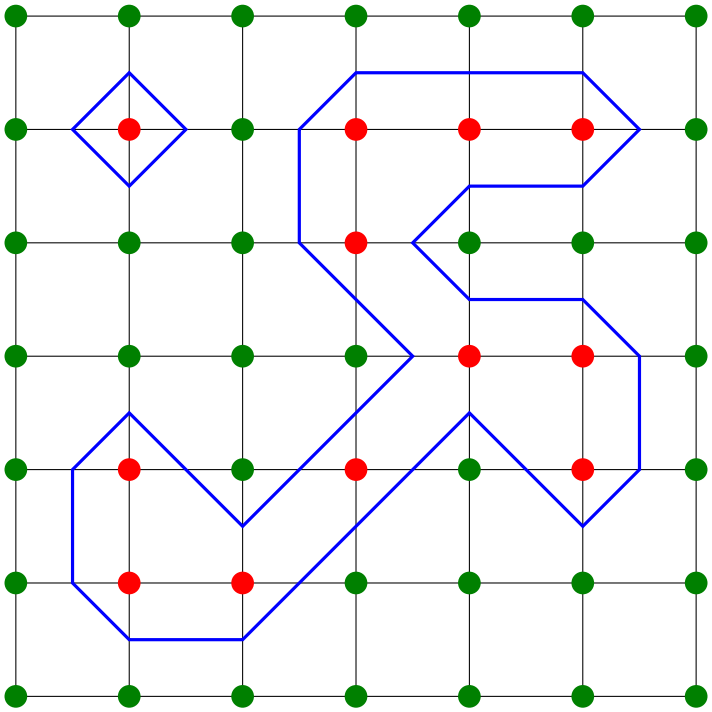
Problem 13 (4 points)

Geometrically subdivide the Bézier curve given by the control points below and at $t = 0.2$ along the curve. Label the new control points A, B, C, ..., G. (The two resulting Bézier curves should share their common point, so only 7 control points are required.)



Problem 14 (3 points)

An implicit surface has been sampled on a regular grid. The nodes have been labeled accordingly (red is negative, green is positive). Sketch out what that implicit surface might have looked like. There is more than one correct answer; suggest something plausible.



Problem 15 (2 points)

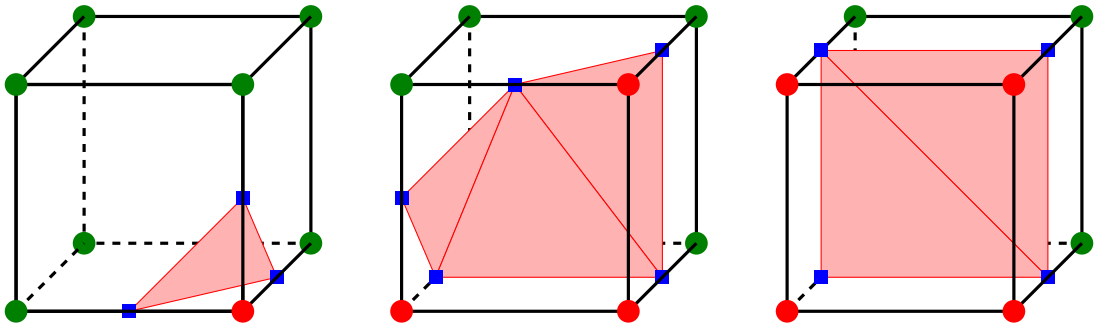
What does the following code do?

```
void foo(int * a, int * b)
{
  while( * a ++ = * b ++ );
}
```

This code copies data from array `B` to `A`, up to and including the first zero entry in the array. If the pointers where of type `char`, this would be an implementation of `strcpy`.

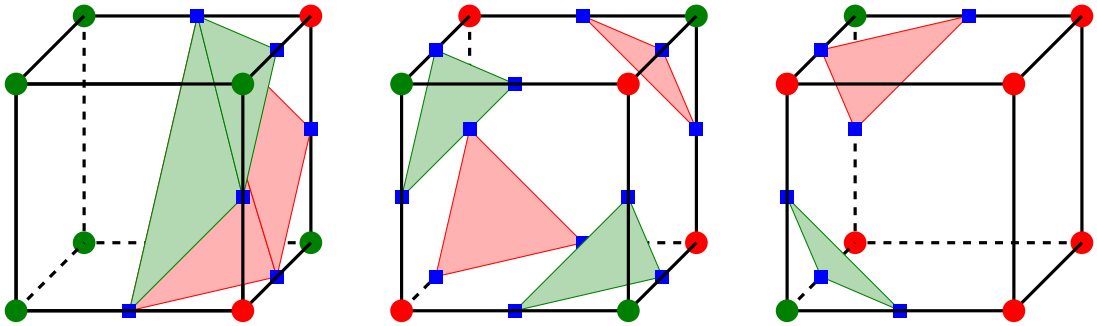
Problem 16 (3 points)

Construct a consistent marching cubes triangulation of the cubes shown below. The cubes should actually be touching, but they have been separated apart for clarity. Draw squares at the vertices of your triangles.



Problem 17 (3 points)

Construct a consistent marching cubes triangulation of the cubes shown below. The cubes should actually be touching, but they have been separated apart for clarity. Draw squares at the vertices of your triangles.



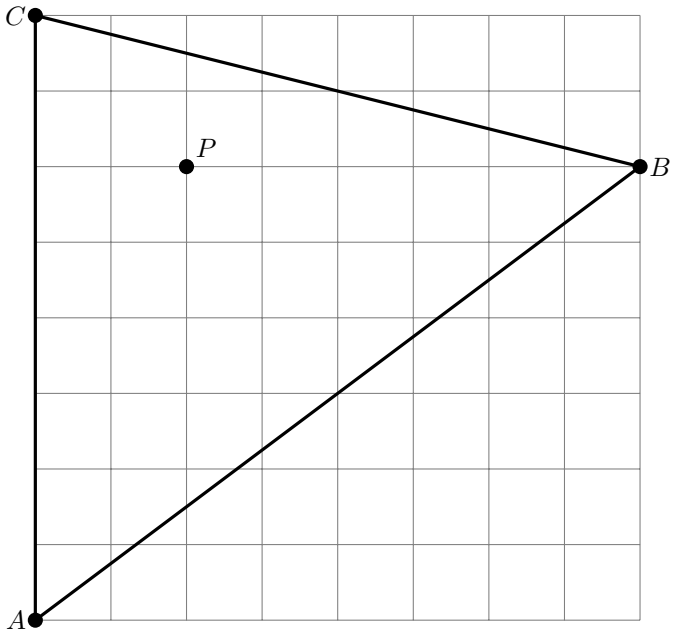
Problem 18 (3 points)

A mesh contains the seven vertices $A - G$ and the six triangles (B, C, A), (C, D, B), (E, D, B), (F, D, C), (D, E, F), (G, E, F). Fix the orientations of the triangles so that they are consistent.

An example of a correctly orientated mesh is (B, C, A), (C, B, D), (E, D, B), (F, C, D), (D, E, F), (G, F, E).

Problem 19 (4 points)

The triangle at right is to be rasterized. The colors of the vertices are A = yellow = $(1, 1, 0)$, B = cyan = $(0, 1, 1)$ and, C = violet = $(1, 0, 1)$. Compute the color of the point P.



To do this, we need to compute the areas of the triangles, which we can then use to compute barycentric weights. Note that all of the triangles we need have an edge that is horizontal or vertical, so $A = \frac{1}{2}bh$ is easy to compute.

$$\text{area}(ABC) = 32$$
$$\text{area}(PBC) = 6$$
$$\text{area}(APC) = 8$$
$$\text{area}(ABP) = 18$$
$$\alpha = \frac{\text{area}(PBC)}{\text{area}(ABC)} = \frac{6}{32}$$
$$\beta = \frac{\text{area}(APC)}{\text{area}(ABC)} = \frac{8}{32}$$
$$\gamma = \frac{\text{area}(ABP)}{\text{area}(ABC)} = \frac{18}{32}$$
$$C_P = \alpha C_A + \beta C_B + \gamma C_C = \left(\frac{24}{32}, \frac{14}{32}, \frac{26}{32}\right) = \left(\frac{3}{4}, \frac{7}{16}, \frac{13}{16}\right)$$

Problem 20 (4 points)

Compute the normal direction for the implicit surface $f(x,y) = 2x^2y^2 - 5xy^2 + 2$ at the point $(2,-1)$. You do not need to simplify your answer.

$$\mathbf{u} = \nabla f = \begin{pmatrix} 4xy^2 - 5y^2 \\ 4x^2y - 10xy \end{pmatrix} = \begin{pmatrix} 4(2)(-1)^2 - 5(-1)^2 \\ 4(2)^2(-1) - 10(2)(-1) \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

$$\|\mathbf{u}\| = 5$$

$$\mathbf{n} = \frac{\mathbf{u}}{\|\mathbf{u}\|} = \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix}$$

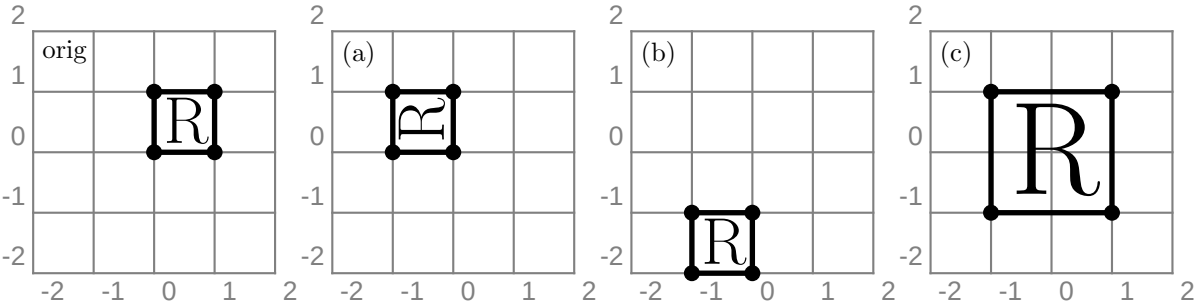
Problem 21 (4 points)

Write a routine that rasterizes a filled circle. That is, it draws pixels that lie on or inside the circle but not pixels that lie outside it. Your routine should be written in C++-like syntax and use the signature `void fill_circle(int x, int y, int r);`, where (x,y) and r are the circle's center and radius. You may call the routine `void draw(int x,int y);` to set the pixel (x,y). You may assume that the circle lies entirely inside the image.

```
void fill_circle(int x, int y, int r)
{
  for(int i=-r;i<=r;i++)
    for(int j=-r;j<=r;j++)
      if(i*i+j*j<=r*r)
        draw(x+i,y+j);
}
```

Problem 22 (6 points)

A square with a letter (shown in the diagram labeled "orig" below) is transformed into each of the configurations (a)-(c). In each case, identify the type of transform and, if possible, find a 3×3 homogeneous transform matrix corresponding to it. In each case, identify the transform as a R=rotation, T=translation, S=uniform scale, X=none of these. R, S, and T can be combined. The most restrictive option should be chosen. Thus, a transform that can be accomplished by a combination of rotation and uniform scale should be described as R+S, not as X.



(a) R. $\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

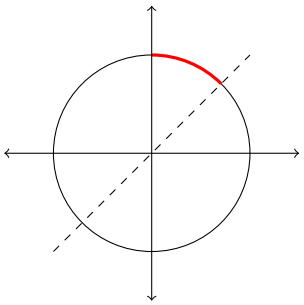
(b) T. $\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix}$

(c) S+T. $\begin{pmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{pmatrix}$

Problem 23 (7 points)

In this problem, we will adapt the midpoint algorithm to rasterize the outline of a circle. Assume first that the circle has integer radius r and is centered at the origin. Lets begin by rasterizing the portion of the circle satisfying $0 \le x \le y$ (shown in red). The general shell for an algorithm like the midpoint algorithm is

```
void rasterize_circle(int r)
{
  int j = first_y;
  for(int i = first_x, stop_criterion; i++)
  {
    draw(i,j);
    if(update_criterion)
      j++; // or j--;
  }
}
```



(a) What values should be used for `first_x` and `first_y`?

`first_x=0;` and `first_y=r;`

(b) Should we use `j++` or `j--`? Why?

`j--;` As we move right (`i++;`) the curve is decreasing, so `j` must decrease.

(c) Suggest something reasonable for `stop_criterion`.

`i <= j;` is a reasonable choice.

(d) For `update_criterion`, we need a function $g(x,y)$ such that $g(x,y) \le 0$ if and only if (x,y) is inside the circle. Suggest a suitable function $g(x,y)$. (You are not allowed to use square roots.)

$g(x,y) = x^2 + y^2 - r^2$

(e) For `update_criterion`, we want to test $g(i+a, j+b)$ at one point $(i+a, j+b)$. What should we use for the constants a and b?

$a = 1$, $b = -\frac{1}{2}$. Recall that we are testing the point halfway between $(i+1, j)$ and $(i+1, j-1)$.

(f) Finally, `update_criterion` will take the form $g(i+a, j+b) < 0$ or $g(i+a, j+b) > 0$. Which inequality do we want? You must justify your answer.

We want $g(i+1, j-\frac{1}{2}) > 0$. We must decrement `j` if the candidate point is too high, which places it outside the circle.

(g) The algorithm above only draws the red portion of the circle. Suggest a simple (and very efficient) modification that can be made to the algorithm above so that it will draw the entire circle.

A pixel location in each of the eight portions of the circle can be deduced from the one pixel computed: `draw(i,j);draw(j,i);draw(i,-j);draw(j,-i);draw(-i,j);draw(-j,i);draw(-i,-j);draw(-j,-i);`