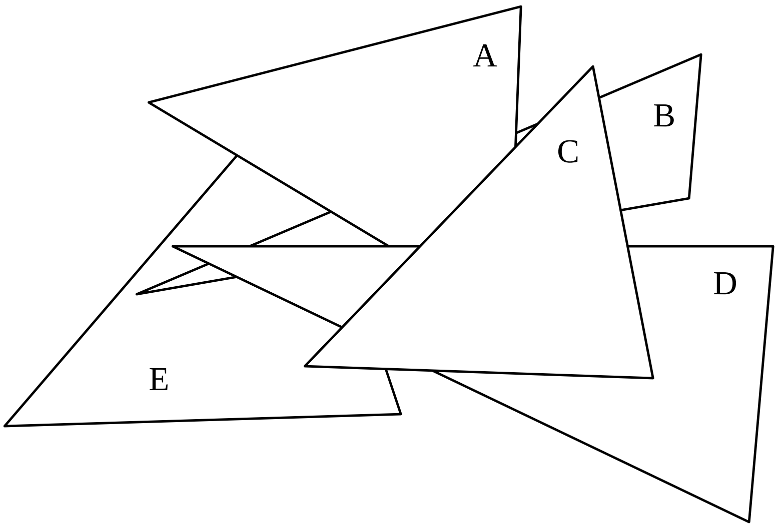
CS450/550 — FINAL EXAM — Spring 1995

Please write your name on each page. This exam is open book, open notes. You have 110 minutes to work the exam. The number of points for each problem is shown in square brackets [110] next to each problem (or part of a problem). One point = one minute.

1. Consider the following scene:



(a) [5] In what order must the triangles in this image be drawn if the Painter's algorithm is used?

(b) [5] Suppose we are using Z buffering. What drawing order would redraw the fewest number of pixels?

2. In depth-sorting, it is necessary to determine for each pair of triangles A and B whether they overlap (when projected onto the viewing plane). We studied three tests: (T1) vertex containment, (T2) edge intersection, (T3) bounding rectangle intersection.

Part (a) [3] List these tests in increasing order of their CPU cost.

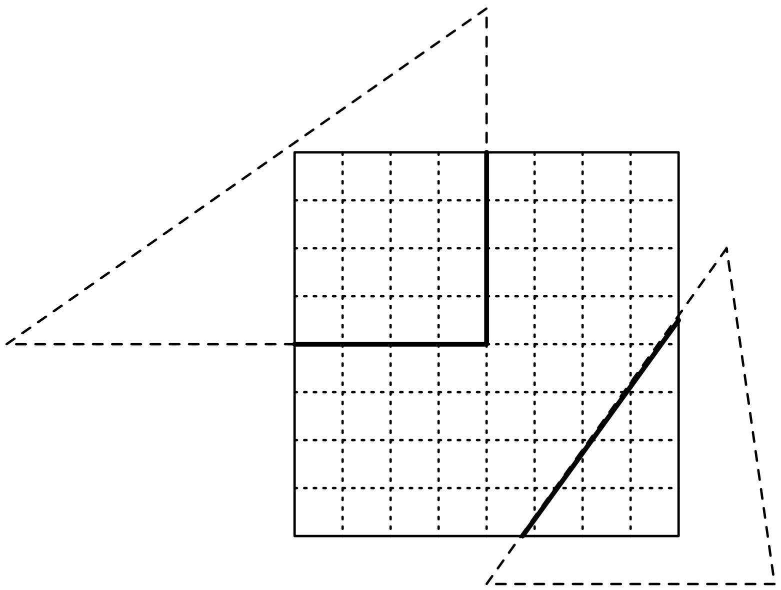
Part (b) [6] Fill in the following table. For each outcome of each test, indicate what we can conclude about whether triangles A and B overlap by writing one of the words "overlap" (if this outcome proves that they overlap), "disjoint" (if this outcome proves that they do not overlap), or "unknown" (if this outcome doesn't tell us whether they overlap). Your answer should apply only to the case where each test is *applied alone—not in combination with other tests.*

	Test Outcome	
Test	True	False
First vertex of one triangle is contained in the other triangle		
Triangle edges intersect		
Bounding rectangles overlap		

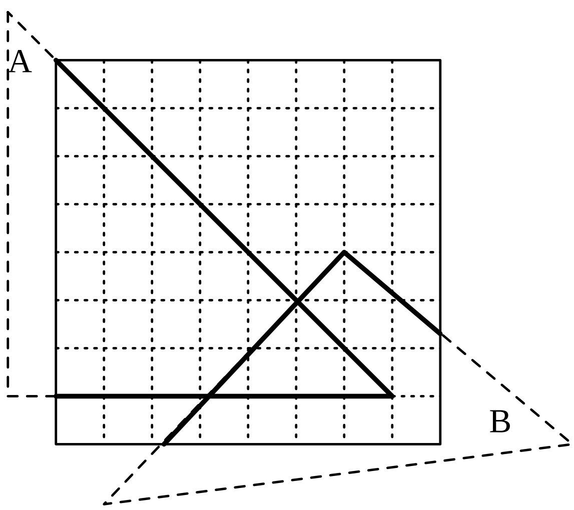
Part (c): [4] Draw a decision tree (or a set of nested `if-then-else` statements) that combines these three tests to decide whether the triangles overlap or are disjoint.

3. Consider the following scenes. In each scene, assume that the Warnock screen subdivision algorithm is applied. When the algorithm subdivides a region, it processes the four subregions in the order top-left, top-right, bottom-left, and bottom-right. Consider each square in these scenes to be a single pixel. The Warnock algorithm fills either individual pixels or rectangular areas. For each scene, draw circles to indicate regions of pixels that are filled as a region. Number the circles to show the order in which they are filled (this is the same notation as we used on Homework 5 and Figure 11.19 p. 426 of the textbook). [Interesting piece of trivia: Warnock is now CEO of Adobe Incorporated.]

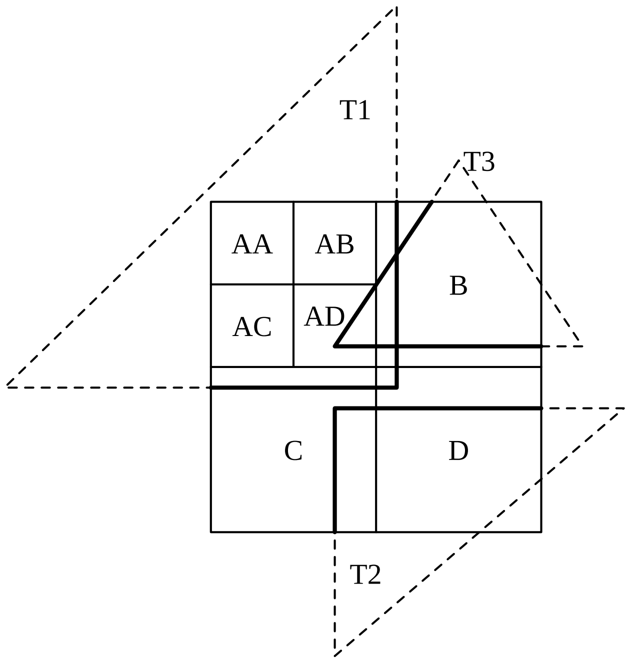
(a) [5]

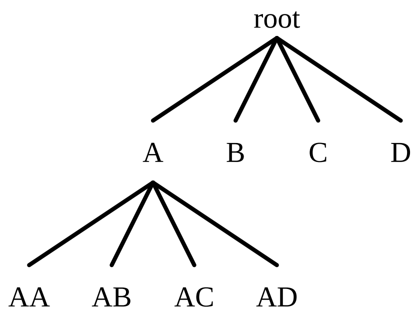


(b) [5] Assume that triangle B is in front of triangle A.

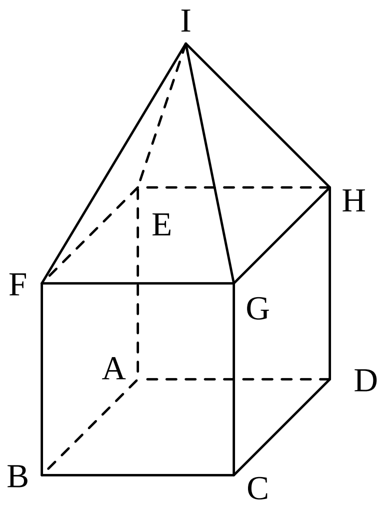


4. [6] In the following scene, assume we are applying the Warnock algorithm. A partial tree of recursive calls is shown on the right, and the regions are labeled to match the nodes of the tree. You may assume that triangle T3 is nearer than triangle T2, which is nearer than triangle T1. What will be the contents of the `ilist`, `slist`, and `dlist` at the start of the recursive call to node **AA**?

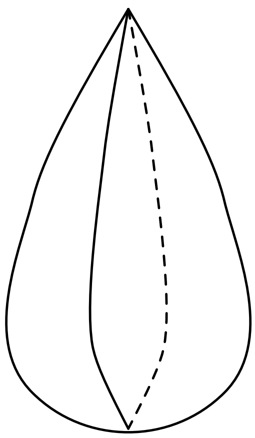




5. Suppose we wish to produce an image of a rain drop using Phong interpolative shading. We want to use the model shown on the left to produce an image that approximates the shape shown on the right.



Model



Desired Image

Part (a) [10] Complete the following tables with the information that would go into a `pmesh` file to be used with Pokorny's C++ code. In the vertex table, just give the name of the vertex (instead of its x/y/z coordinates). To "clone" a point, enter that vertex name more than once.

Vertexes

Number of vertexes: ______________

Index	Name
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	

Faces

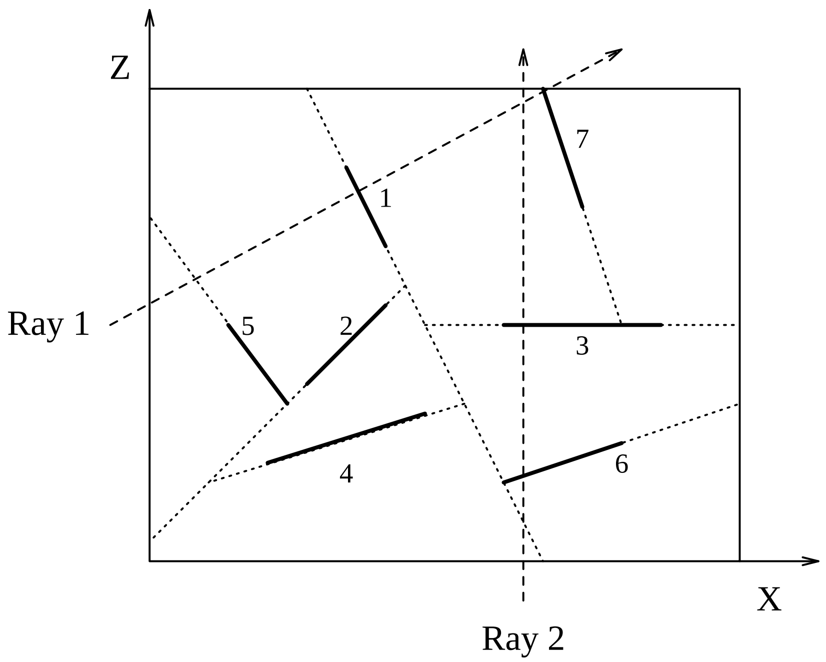
Number of faces: ______________

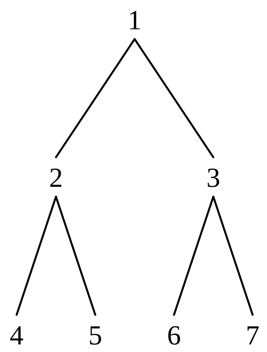
Index	Vertexes
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	

Part (b) [5] What is the biggest problem with using Phong shading to model a rain drop? Briefly describe how you would change this model to create a better image of a water drop (still using Phong shading).

6. [6] What information is needed to compute the illumination at a point P on an object using Phong's rule?

7. Consider the following 2-D world and an associated BSP tree to be used for ray tracing. The front side of each splitting line is in the $-Z$ direction. The child on this side is the left child. For each ray, indicate which line segments will be tested for intersections with the rays (and in what order).





(a) [5] Ray 1:

(b) [5] Ray 2:

8. Consider the following world and device coordinate systems.

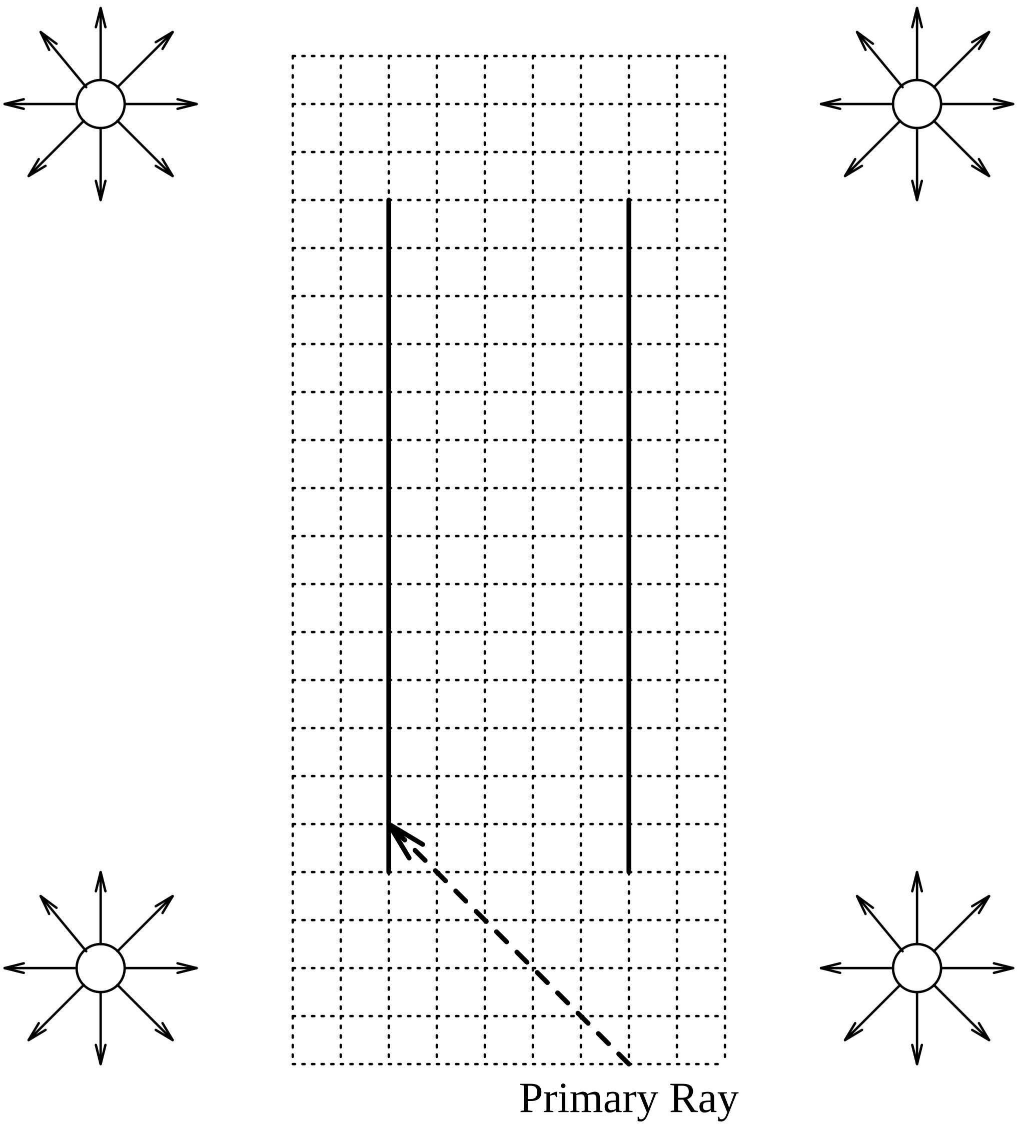
X-window: 100 by 100.
World coordinates of X-window: $left = -1, right = +1, bottom = -1, top = +1, z = 0$.
Location of eyepoint: $(0, 0, -1)$

(a) [3] Compute the direction vector of a ray that starts at the eye point and passes through the pixel with device coordinates (0,0):

(b) [3] Compute the direction vector of a ray that starts at the eye point and passes through the pixel with device coordinates (50,50):

(c) [3] Compute the direction vector of a ray that starts at the eye point and passes through the pixel with device coordinates (50,0):

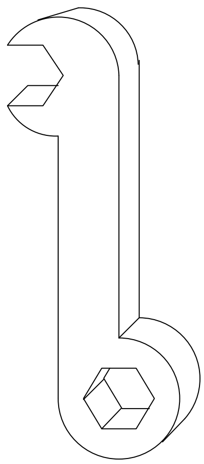
9. The following scene shows one primary ray, four point light sources, and two vertical mirrors in a 2D world. The mirrors are reflective on all sides. They are not quite perfect—they exhibit some diffuse reflection as well. The ray strikes the first mirror at a 45 degree angle.



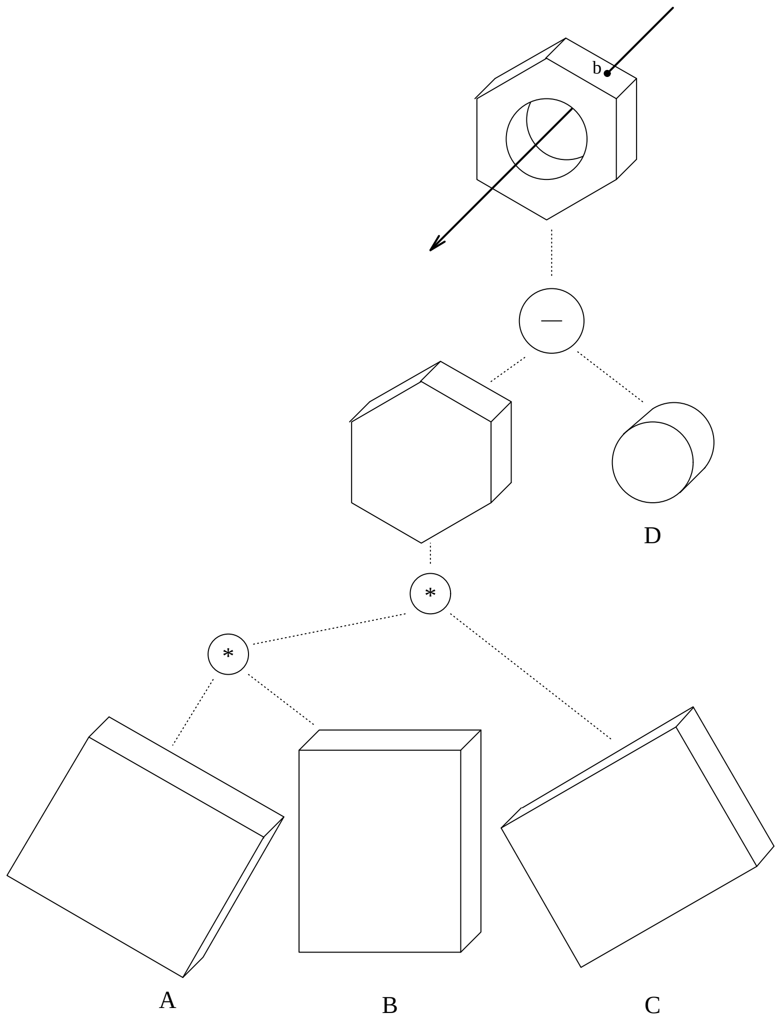
(a) [5] How many additional rays will need to be traced to finish tracing this primary ray?

(b) [5] Draw those rays on the figure. The dotted grid lines are there to help you; they are not part of the scene itself.

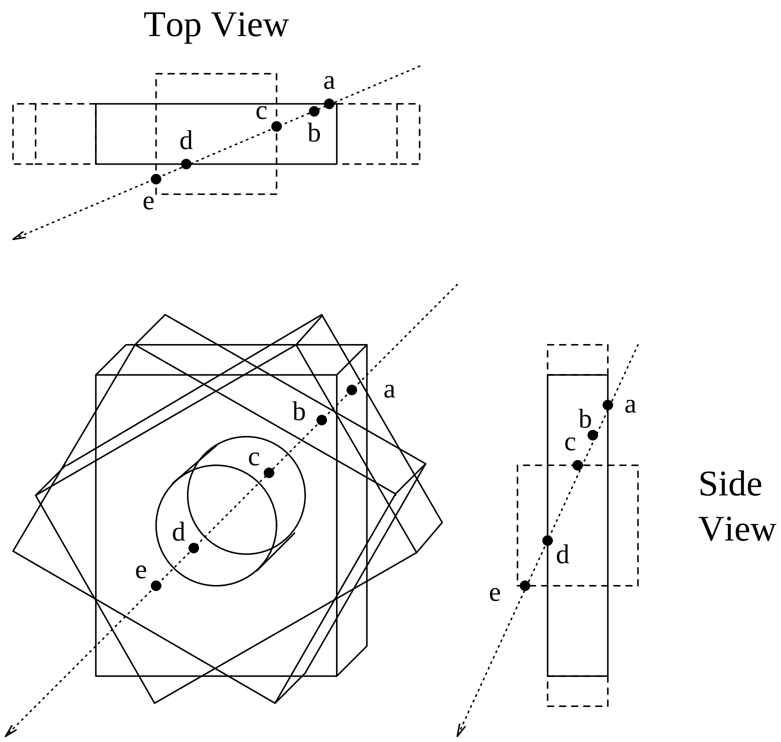
10. [10] Consider the following wrench. Show a Constructive Solid Geometry tree that will produce this object. You may assume you that you have a hexagonal prism available as a primitive.



11. Consider the following CSG tree for a hexagonal nut. The hexagonal prism is constructed by taking the intersection of three blocks. A cylinder is then subtracted out. A ray is shown piercing the nut (entering at point b).



The following figure gives three views of the ray passing through the four primitive objects.



(a) [5] Give the hit lists constructed for each of the four CSG primitives.

(b) [3] Show the combined hit list for the intersection of objects A, B, and C.

(c) [3] Show the combined hit list for the whole nut.