CSC418/2504F: Midterm Test: SAMPLE

Wednesday,

Family Name:________________________________

First Name: ________________________________

Student ID: ________________________________

Instructions:

Attempt **all** questions.
There are five questions.
The total mark is 24.

You have 60 minutes to complete the test.

Aids allowed: Calculators
Textbooks and notes are NOT allowed.

1:	/5
2:	/8
3:	/2
4:	/12
Total:	/27

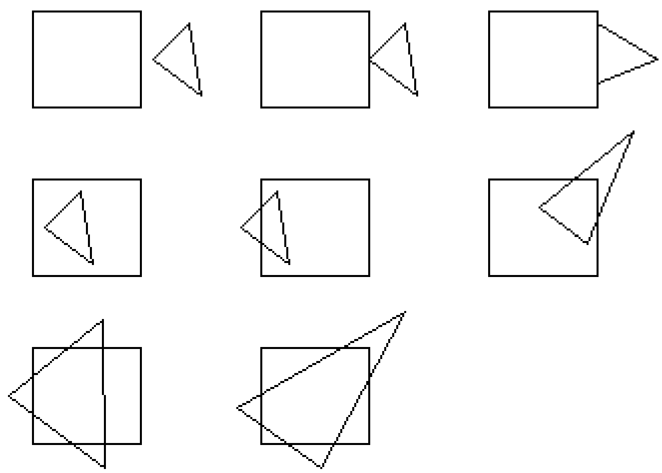
1. Clipping and scan conversion [5 marks]:
   a. [2 marks] List the pixels produced by scan-converting the line segment from (0,4) to (6,0).

(0,4)(1,3)(2,3)(3,2)(4,1)(5,1)(6,0)

   b. [3 marks] Suppose we clip a polygon with **n** sides against a rectangular window. The clipped polygon will have **m** sides, and often **n** ≠ **m**.

      If **n** = 3, what values can **m** have? Give a sketch for each value of **m**.

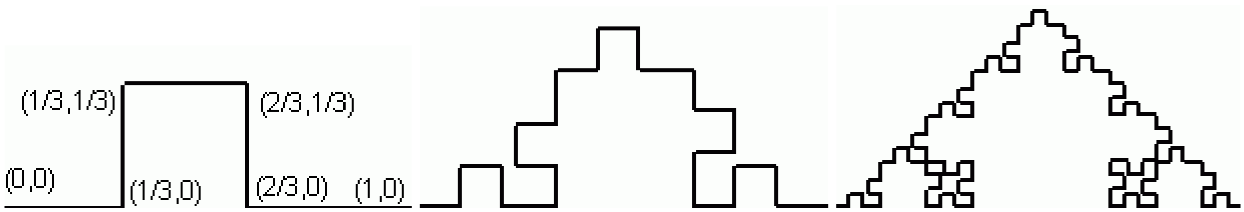
- **m can range from 0 to 7. Here is a sketch for each case:**



2. Modeling: [8 marks] For this question, use the following openGL-like function calls:

trans(t_x,t_y)	scale(s_x,s_y)	rot_z(ang)
glBegin()	push()	pop()

In this question, you will write code to draw a fractal "meander".



Order 1 Order 2 Order 3

a. [3 marks] Write code to draw an order-1 meander.

```
order1 () {
  glBegin(GL_LINE_STRIP);
  vertex(0,0);
  vertex(0.3333,0);
  vertex(0.3333,0.3333);
  vertex(0.6666,0.3333);
  vertex(0.6666,0);
  vertex(1,0);
  glEnd();
}
```

c. [5 marks] As you can see, an order-1 meander consists of 5 **segments**, each length 1/3. If we **replace each segment** with a whole **meander scaled down by 1/3**, we obtain a higher-order meander. Write a function **meander(int n)** which draws an order-**n** meander. It is easiest to define the function recursively.

```
meander (int n) {
  if (n == 1) {
    order1();
  }
  else {
    push();
      scale(0.3333,1);
      meander(n-1);
    pop();
```

```
      trans(0.3333,0);
      rotz(90);

      push();
        scale(0.3333,1);
        meander(n-1);
      pop();
      trans(0.3333,0);
      rotz(-90);

      push();
        scale(0.3333,1);
        meander(n-1);
      pop();
      trans(0.3333,0);
      rotz(-90);

      push();
        scale(0.3333,1);
        meander(n-1);
      pop();
      trans(0.3333,0);
      rotz(90);

      push();
        scale(0.3333,1);
        meander(n-1);
      pop();
    }
  }
```

3. True or False with a reason: [2 marks]
   a. [1 mark] Real cameras use a lens to compensate for the fact that apertures larger than a pin-hole make the image blurry.

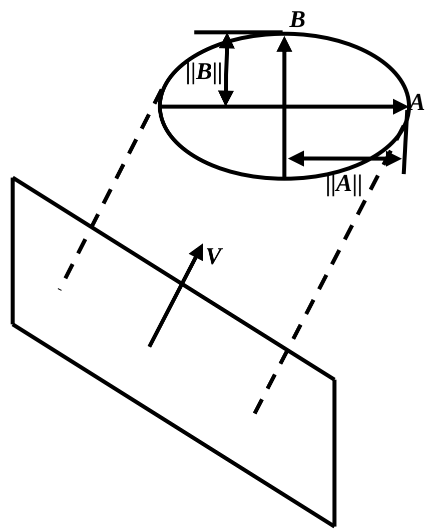
True. Lenses cause rays through the aperture to focus towards a smaller region on the image plane making the image sharper.

   b. [1 mark] The result of multiple rotations to a point is order dependent but combinations of scaling and translations can be applied in any order with the same result.

False. Take a point at the origin and scale it 2x and translate it to (5,5).

If translated and then scaled the resulting point is (10,10).

4. Projection, drawing primitives: [12 marks] An ellipse defined by semi-major and semi-minor axes vectors ***A, B*** is to be drawn using orthographic projection with view direction ***V*** :



a. [4 marks] Explain intuitively why the projected curve must also be an ellipse or a straight line.

Possible answers (you will get partial credit for a reasonable attempt at an answer):

The view space transform is a change of basis rotate and translate transform that leaves an equation in x,y,z to have the same or lesser degree. Orthographic projection sets z=0. A quadratic thus projects to a quadratic or linear equation in x,y. If quadratic the ellipse being closed projects to a closed quadratic that is an ellipse. (2 marks).

When ***(AxB).V=0*** or the plane of the ellipse is perpendicular to the view plane the curve projects to a line. (1 mark),

Ellipse eq. x=a*cost, y=bsint z=0. Any view space rotation can be seen as a sequence of 3 rotation around x,y,z axes. Ignore rotation of ellipse around z axes which is the projection plane (ellipse rotated in 2D is an ellipse). After rotation around y by angle m, eq. is x'=x, y'=ycosm, z'=xsinm. After rotation around y by angle n eq. is x'' =xcosn – xsinm*sinn, y''=ycosm , z''=xsinn+xsinm*cosn. Throw away z'' in the equation. We are left with the projected curve x''= a(cosn-sinm*sinn)cost, y''= b(cosm)sint which is still the parametric eq. of an ellipse. (2 marks).

b. [4 marks] Under what conditions can the projected curve be a circle and what is the radius of that circle as a function of ***A, B, V.***

A'=sqrt($||A||^2$-$(A.V)^2$). B'=sqrt($||B||^2$-$(B.V)^2$). The curve is a circle when A'=B'=radius of the circle.

(1 mark for getting the A', B' values, 1.5 marks for saying that when the new semi-maj, semi-minor axes are equal we get a circle, 0.5 marks for pointing out that the radius is A' or B'.

c. [4 marks] Write pseudo code for a loop that plots the ellipse defined as $(x/a)^2 + (y/b)^2 = 1$ as a sequence of *n* line segments.

```
ang=2pi(n-1)/n;px=a*cos(ang);py=b*sin(ang);

For (i=0;i<n;i++) {ang=2pi*i/n; x=a*cos(ang);y=b*sin(ang); line(x,y,px,py);x=px;y=py;}
```

(1 mark for using the parametric eq. of ellipse, 1 mark for loop str., 1 mark for getting the angle incr. right). Same marking scheme if you use the explicit equation for 4 quadrants.

d. EXTRA CREDIT [2 marks] What property of the curve should determine where to sample points on the ellipse so that the *n* line segments that connect them best approximate the ellipse. What part of your pseudo-code in *part c* needs to change to reflect this.

The arc length of the ellipse between sample points should be roughly equal for better approximation. Curvature of the ellipse can be used as a measure of this. (1.5 marks). The angle increment is now not fixed but depends on the arc length between samples (0.5 marks).