NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 1, 2010/2011

CS3241 - COMPUTER GRAPHICS

Nov 2010 Time Allowed: 2 Hours

INSTRUCTION TO CANDIDATES

1. This is an OPEN book examination.
2. This examination paper contains **SIX (6)** questions and comprises **EIGHT (8)** printed pages.
3. Answer **ALL** questions within the spaces provided in this booklet.
4. You are allowed to use the back of the paper but please remember to state "P.T.O."
5. ***Cross out any draft*** or otherwise we will mark the poorer answers.
6. No calculator should be needed in this exam. You can leave your answer in surd form, namely, you can write $\sqrt{2}$ instead of 1.4142...
7. In the programming questions, you can assume some basic normal vector arithmetic functions are provided for the 2D vector class `Vector`.
8. Please write your matriculation number below, but NOT your name.

TIDINESS COUNTS!

We will deduct marks if your writing is too messy.

MATRICULATION NUMBER:_____

(this portion is for examiner's use only)

Question	Max. Marks	Score	Check
Q1	10		
Q2	6		
Q3	17		
Q4	7		
Q5	4		
Q6	6		
Total	50		

Question 1 [10 marks]

You are given 10 statements below. State if each statement is True (T) or False (F) by writing your answer in the box provided. A correct answer will give you 1 mark. An empty answer will give you zero, but ***a wrong answer will result in -1 mark***. (Lowest mark for this question is 0)

	Clipping is performed after projection transformation
	OpenGL library works for PC (Windows) and MacOS only.
	In OpenGL, the funciton `glNormal3f()` is for normalizing a vector to it's unit length.
	If we increase the resolution of the display device to be double finer or higher than that human eyes can handle, we can solve the alisaing problem.
	If we perform rotation and then scaling, the result may be different if we exchange their orders.
	Texture mapping and environmental mapping techniques can be applied to both realtime graphics(SCA) and ray-tracing.
	It is more difficult to find the intersection of a straight line with Bezier patches than a quadratic function $f(x,y,z) = 0$.
	All C1 continuous Bezier curves are G1 continuous.
	We need to know the position of the viewer to calculate diffuse reflection.
	The whole Bezier patch is contained in the convex hull of its control points.

Question 2 [6 marks]

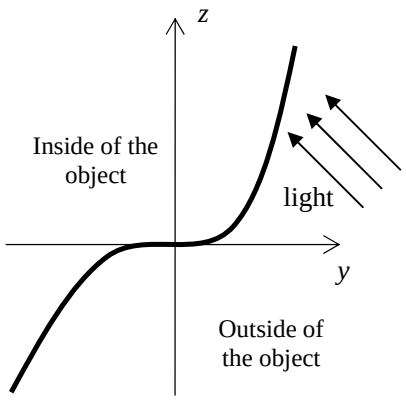
Fill in the code for setting up the normals for the following C++ codes to draw a sphere for both smooth and flat shadings:

```
void drawSphere(double r)
{
  glDisable(GL_NORMALIZE);  // Disable the automatic normalization of normal vectors
  int i,j;  int n = 20;  double x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4;
  for(i=0;i<n;i++)
    for(j=0;j<2*n;j++)
      if(m_Smooth)  { // Smooth shading code here
        x1 = sin(i*M_PI/n)*cos(j*M_PI/n);
        y1 = cos(i*M_PI/n)*cos(j*M_PI/n);
        z1 = sin(j*M_PI/n);
        x2 = sin((i+1)*M_PI/n)*cos(j*M_PI/n);
        y2 = cos((i+1)*M_PI/n)*cos(j*M_PI/n);
        z2 = sin(j*M_PI/n);
        x3 = sin((i+1)*M_PI/n)*cos((j+1)*M_PI/n);
        y3 = cos((i+1)*M_PI/n)*cos((j+1)*M_PI/n);
        z3 = sin((j+1)*M_PI/n);
        x4 = sin(i*M_PI/n)*cos((j+1)*M_PI/n);
        y4 = cos(i*M_PI/n)*cos((j+1)*M_PI/n);
        z4 = sin((j+1)*M_PI/n);

        glBegin(GL_POLYGON);


          glVertex3d(r*x1,r*y1,r*z1);


          glVertex3d(r*x2,r*y2,r*z2);


          glVertex3d(r*x3,r*y3,r*z3);


          glVertex3d(r*x4,r*y4,r*z4);

      glEnd();
      } else {         // Flat shading code here







        glBegin(GL_POLYGON);


          glVertex3d(r*x1,r*y1,r*z1);


          glVertex3d(r*x2,r*y2,r*z2);


          glVertex3d(r*x3,r*y3,r*z3);


          glVertex3d(r*x4,r*y4,r*z4);


        glEnd();
      }
}
```

Question 3. [17 marks]

An object with equation $z = x^3+y^3$ is illuminated with a parallel white light source with direction vector $\boldsymbol{L}$ = [0, -1, 1] as indicated by the arrows in the figure, *i.e.* $\boldsymbol{L}$ is parallel to the *yz*-plane.

a) If we only consider the part of the paraboloid at the plane of $x = 0$. It will be a parabola with equation $z = y^3$. Compute the tangent vector at a point on the object with respect to *y* on this plane. **[3 marks]**



Answer:

b) What is the normal vector (the one which is pointing 'out' of the object) at a point on the parabola with respect to *y*? **[2 marks]**

Answer:

c) The surface of the paraboloid exhibits only **diffuse reflection** (i.e. no ambient and specular reflections). Based on the Phong model, find the point on the surface that is brightest to a viewer and explain your answer. **[3 marks]**

Answer:

Question 3 (cont.)

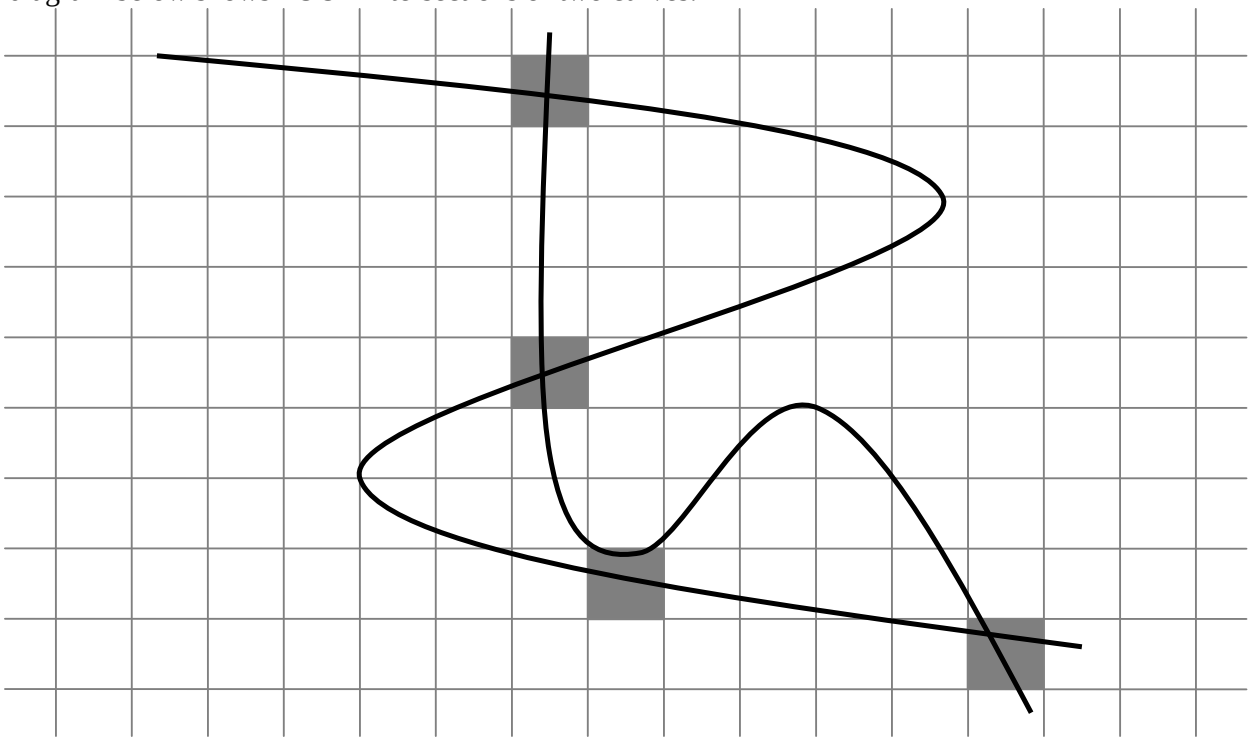
d) A viewer is at the position (0,-0.5,-0.5) looking at the direction (0,1,1). What is the ray equation for this viewer? [**2 marks**]

e) Find all the intersections (the positions) and how many are there? **[5 marks]**

f) Which one is the first point that the viewer can see? **[2 marks]**

QUESTION 4 [7 marks]

Given two cubic Bezier curves, $p(t)$ (with control points $p0$, $p1$, $p2$ and $p3$) and $q(t)$ (with control points $q0$,$q1$,$q2$ and $q3$). We deem that if the two curves are on the same pixel, then they "intersect" each other. The diagram below shows FOUR intersections of two curves.



We want to find ALL of their intersections. (But you do NOT have to compute self-intersection, as in $p(t)$ intersects $p(t)$ itself) Write a **recursive** function using the **divide-and-conquer method** and store all pixels with the intersection into the set `SetOfPixels`, assuming it is empty in the beginning and it is a global variable. You can add a pixel by `SetOfPixels.add(pixel)` and assume that it is able to remove duplication if a pixel is being added twice.

You can assume the following functions are implemented and given:

`TestTwoConvexHulls(Vector p0, Vector p1, Vector p2, Vector p3, Vector q0, Vector q1, Vector q2, Vector q3)`
Return TRUE if the convex hulls of {p0,p1,p2,p3} and {q0,q1,q2, q3} overlap each other. Return FALSE otherwise.

`TestHullSmallerThanPixel(Vector p0, Vector p1, Vector p2, Vector p3)`
Return TRUE if the convex hull of {p0,p1,p2,p3} is within 1 pixel. Return FALSE otherwise.

`PixelOfCH(Vector p0, Vector p1, Vector p2, Vector p3)`
Return the pixel containing the convex hull of {p0,p1,p2,p3}.

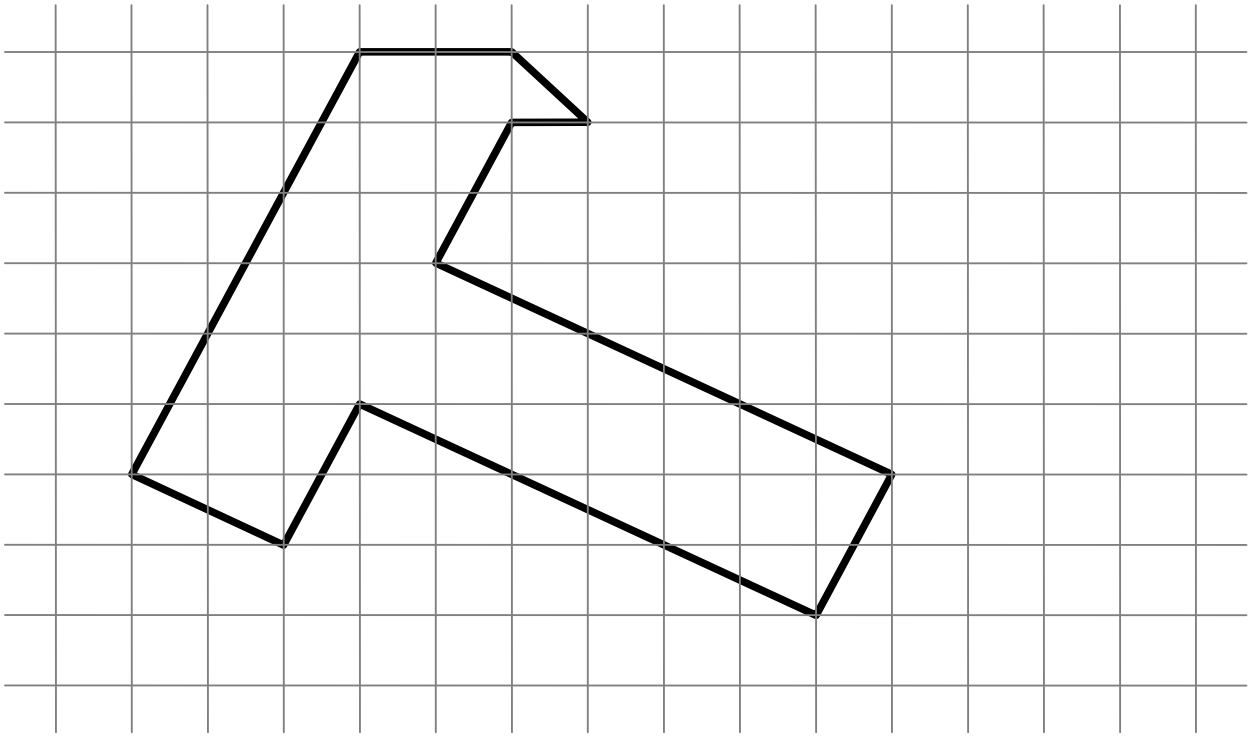
And also the class `Vector` and its operation, e.g. you can simply write `3*p0` if `p0` is a `Vector`.
If you assume any other simple function such as `max(…)` for returning the maximum of a set of numbers, please state so by comments.

QUESTION 4(Cont.)

```
ComputeAllIntersections(Vector p0, Vector p1, Vector p2, Vector p3,
                        Vector p0, Vector p1, Vector p2, Vector p3)
{

    Vector p10,p11,p12, p20,p21, p30;
    Vector q10,q11,q12, q20,q21, q30;









}
```

Question 5 [4 marks]

Given the outline of a polygon below and assume it is solid black in color. Imagine that you have to scanline convert the picture and shade the pixels with anti-aliasing effect with a box filter. Write down the degree of "blackness" of each pixel assuming the full blackness is one hundred. (You do not have to color each pixel, just write the number in the box is enough)



Question 6 [6 marks]

Suggest the diffuse and specular reflective properties for the materials for the following objects. Your answers do not need to be very exact. However, they should show differences between the three objects.
Answers:

a) A squash ball (dull green color)

	Red	Blue	Green
Diffuse (0.0~1.0)			
Specular (0.0~1.0)			
Shininess Coefficient: n =			

b) A silver ring

	Red	Blue	Green
Diffuse (0.0~1.0)			
Specular (0.0~1.0)			
Shininess Coefficient: n =			

c) Polished red leather shoes

	Red	Blue	Green
Diffuse (0.0~1.0)			
Specular (0.0~1.0)			
Shininess Coefficient: n =			

- END OF PAPER -