The first part of the lab involved importing data to yield a mesh image of a barn. An animation was created using the delay function `SDL_Delay()` of the SDL library. The following image was the final result:

![Mesh Image](image_url)

The second part required the enabling of the face culling feature for all the back faces in the mesh image. This was accomplished through `glEnable(GL_CULL_FACE)` and `glCullFace(GL_BACK)`. The following image is the result:

![Face Culling Image](image_url)
The third part modified the image from a filled to lined polygonal shapes. This was done by changing the shape drawing construct from the GL_POLYGON to the GL_LINE_LOOP parameter options. The modification resulted in the following:
The fourth part required the drawing of another object using the mesh technique. The image created as a barn with four-faced pyramidal roof, as the following:
The final code, with the additions and modifications (labeled in red), is provided below:

/****************************
**Name: Benjamin Benjamin
**Course: CSE-621
**Term: Spring 2013
**Description: mesh.cpp -- a program that imports vertex, polygonal
face and normal vector information to yield a three dimensional
looking mesh image.
**************************/
#include "mesh.h"
#include "SDL/SDL.h"

using namespace std;

Mesh::Mesh() //constructor
{
    numVerts = numFaces = numNormals = 0;
    pt = NULL;
    norm = NULL;
    face = NULL;
}
bool Mesh::isEmpty() {
    return (numVerts == 0) || (numFaces == 0) || (numNormals == 0);
}

void Mesh::setColor( int n ) {
    if ( n == 1 )
        glColor3f( 1, 0, 0 );
    else if ( n == 2 )
        glColor3f( 0, 1, 0 );
    else if ( n == 3 )
        glColor3f( 0, 0, 1 );
    else if ( n == 4 )
        glColor3f( 1, 1, 0 );
    else if ( n == 5 )
        glColor3f( 1, 0, 1 );
    else if ( n == 6 )
        glColor3f( 0, 1, 1 );
    else if ( n == 7 )
        glColor3f( 1, 0.5, 0 );
    else if ( n == 8 )
        glColor3f( 0.5, 0.5, 0.5 );
    else
        glColor3f( 0, 0, 0 );
}

void Mesh::drawMesh() {
    // use OpenGL to draw this mesh
    // draw each face of this mesh using OpenGL: draw each polygon.
    if( isEmpty() ) return; // mesh is empty
    glEnable( GL_CULL_FACE);
    glCullFace( GL_BACK);
    for(int f = 0; f < numFaces; f++) // draw each face
    {
        glBegin(GL_POLYGON);  // To draw unfilled polygons use
        cout << endl;
        setColor( f );
        for(int v = 0; v < face[f].nVerts; v++) // for each vertex
        {
            int in = face[f].vert[v].normIndex; // index of this normal
            int iv = face[f].vert[v].vertIndex; // index of this vertex
            glNormal3f(norm[in].x, norm[in].y, norm[in].z);
            cout << "[ " << norm[in].x << ", " << norm[in].y << ", " << norm[in].z << "]" << " ";
            glVertex3f(pt[iv].x, pt[iv].y, pt[iv].z);
        }
        glEnd();
    }
}

Name: Benjamin Benjamin
Course: CSE-621
Term: Spring 2013
Assignment: Lab #3
cout << "\"(\" << pt[iv].x << "," << pt[iv].y << "," << pt[iv].z << ")\"; 
}
glEnd();
SDL_Delay(1000);
glFlush();
cout << endl; 
} //drawMesh

//read Mesh data from file
int Mesh::readFile(char * fileName)
{
    fstream infile;
infile.open(fileName, ios::in);
cout << "opening file " << endl;
if(infile.fail()) return -1; // error - can't open file 
if(infile.eof())  return -1; // error - empty file
infile >> numVerts >> numNormals >> numFaces;
pt = new Point3[numVerts];
norm = new Vector3[numNormals];
face = new Face[numFaces];
//check that enough memory was found:
if( !pt || !norm || !face) return -1; // out of memory 
cout << "file open O.K." << endl;

for(int p = 0; p < numVerts; p++) // read the vertices
    infile >> pt[p].x >> pt[p].y >> pt[p].z;
for(int n = 0; n < numNormals; n++) // read the normals
    infile >> norm[n].x >> norm[n].y >> norm[n].z;
cout << "numFaces = " << numFaces << endl;
for(int f = 0; f < numFaces; f++)// read the faces
{
    infile >> face[f].nVerts;
    face[f].vert = new VertexID[face[f].nVerts];
    for(int i = 0; i < face[f].nVerts; i++)
        infile >> face[f].vert[i].vertIndex;
    for(int i = 0; i < face[f].nVerts; i++)
        infile >> face[f].vert[i].normIndex;
}
return 0; // success
} //readFile

Evaluation: All parts of the lab assignment were completed by providing concise description of the code written and screenshots taken.
Grade: 20/20.