# UM EECS 487 Lab 0

*17 problems totaling 35 points.*

Due: Monday, Sep. 8th, 2014 in lecture,
Turn in Part I in hardcopy by 9:40 am

Review the grading policy page on the course web site.

You are allowed to consult both online and offline sources, including humans, to help solve these problems. If you do consult outside sources, i.e., other than yourself and the teaching staff, you MUST cite them. You do not need to cite the teaching staff nor the textbooks nor the lecture notes. These are the only exceptions. What you turn in must be your individual work. Your classmates can give you an idea on how to approach a problem, but they cannot give you any solution to these problems. If you find an online solution to any of these problems, you are allowed to consult them, but not to use them verbatim. You must phrase your solution in such a way that shows you have understood the problem and solution. Violation of any part of the above policy will be a violation of the Honor Code. If you turn in a handwritten solution, please write legibly. Illegible scribble will earn zero points.

To incorporate publicly available code in your solution is considered cheating in this course. To pass off the implementation of an algorithm as that of another is also considered cheating. For example, if the assignment asks you to implement sort using heap sort and you turn in a working program that uses insertion sort in place of the heap sort, it will be considered cheating. If you can not implement a required algorithm, you must inform the teaching staff when turning in your assignment, e.g., by documenting this in your writeup.

## 1. Linear Algebra

For this part you may want to first read the *Vectors and Matrices* course note available on the course website. This course uses a fair amount of linear algebra and thus it is extremely important that you understand all of the material involved in that note.

1.) (2 pts) **Angle Between Vectors.** What is the angle, in radians, between $\vec{u} = \langle 1, 2, 3 \rangle$ and $\vec{v} = \langle -5, 8, -2 \rangle$? What is that angle in degrees?

2.) (2 pts) **Linear Equations With Matrices.** Given the following system of equations

$$\begin{aligned} 2x + 6y + 2z &= -2 \\ x + 9y + 3z &= 1 \\ -3x - 3y + z &= -1, \end{aligned}$$

find all solutions or demonstrate that none exist. In order to solve this, set up the constraint matrix and then compute and use its inverse (by hand, or **no credit**) to find the solution(s) or show that there are none.

3.) (2 pts) **Projection.** Given the two vectors $\vec{u} = \langle -3, 2, 4 \rangle$ and $\vec{v} = \langle 2, 0, -1 \rangle$, find the component of $\vec{u}$ orthogonal to $\vec{v}$ (**not** the component of $\vec{v}$ orthogonal to $\vec{u}$). It may help to draw a picture for this question.

4.) (2 pts) **Triangle Area.** Given the two vectors $\vec{u} = \langle -3, 2, 4 \rangle$ and $\vec{v} = \langle 2, 0, -1 \rangle$, find the area of the triangle with edges $\vec{u}, \vec{v}$, and $\vec{u} - \vec{v}$.

5.) (2 pts) **Matrix Multiplication.** Find $\hat{M}\hat{N}$ and $\hat{N}\hat{M}$, given $\hat{M} = \begin{pmatrix} 1 & 7 & -3 \\ 2 & 2 & -4 \\ 1 & 0 & 7 \end{pmatrix}$ and $\hat{N} = \begin{pmatrix} -5 & 6 & 10 \\ 3 & 5 & -7 \\ 6 & 4 & 2 \end{pmatrix}$. Show all work and triple check the answers; there will be no partial credit for this question, and no credit if work is not shown.

6.) (2 pts) **Vector (as a Matrix) Multiplication.** Given the two vectors $\vec{u} = \langle -3, 2, 4 \rangle$ and $\vec{v} = \langle 2, 0, -1 \rangle$, find $\vec{u}^T\vec{v}$ and $\vec{u}\vec{v}^T$. If the answer does not exist, explain why.

7.) (2 pts) **Plane.** Let there be three points: $\vec{u} = \langle 1, 2, 3 \rangle, \vec{v} = \langle 3, -2, 7 \rangle$, and $\vec{w} = \langle -2, -2, 4 \rangle$. Find the equation of the plane containing them in the form $f(x, y, z) = 0$.

## 2. Intro to OpenGL and GLUT

Open `warmup.cpp` and examine it.

1.) (2 pts) **Point Drawing.** Locate the three instances of `/* Task 1: ... */`. Set the global variable `mode` to the correct OpenGL draw mode for drawing points in two instances, observe how `mode` is used in the other instance.

2.) (2 pts) **Line Drawing.** Locate `/* Task 2: ... */` and set the global variable `mode` to draw a line instead. You can then toggle the app between drawing a line and drawing 2 points.

3.) (2 pts) **Line Thickness.** Locate `/* Task 3: ... */` and set the line thickness to 3 pixels, the default is 1.

4.) (2 pts) **Quadrilateral (Square) Drawing.** Locate `/* Task 4: ... */`. Make it so that a red square is drawn at each end of the line. Each square should be centered on an endpoint and have a side length of `2*ENDHWIDTH`.

5.) (2 pts) **Draw Coordinate Axes.** Locate `/* Task 5: ... */`. Draw a green line along the x- and y-axes. Be sure that the axes stretch from edge to edge on the screen, i.e., not just along the positive portions of the axes are shown. After you completed the next task, you will see that the axes move up and to the right.

6.) (2 pts) **Simple "Animation".** Locate `/* Task 6: ... */`. The provided function `refresh()` shifts the viewport, providing a simple "animation." Register this function to be called when the system is idle. You'll need to hit '`a`' to toggle the animation on and off.

7.) (2 pts) **Polygon Mode.** Locate `/* Task 7: ... */`. Set the polygon drawing mode to draw lines for the front face instead of filling the polygon.

8.) (2 pts) **Attribute Stack.** Locate `/* Task 8: ... */`. Draw a filled blue square centered at the origin with side length 10. Only this blue square at the origin should be filled; the squares at the endpoints of the line should remain drawn in outline instead of filled.

9.) (3 pts) **Mouse and Cursor.** Locate `/* Task 9: ... */`. Register the provided `cursor()` function as a GLUT call back for handling mouse cursor (passive) moved events. Once registered, you should see the cursor changing to a cross hair whenever it gets near one of the two end points. Locate `/* Task 10: ... */`. Register the provided `mouse()` function as a GLUT call back for handling mouse button presses. Once registered, you should be able to click on either end points of the line and drag it to a new location. Note that the line is redrawn only when you release the mouse button. Now locate `/* Task 11: ... */`. Register the provided `drag()` function as a GLUT call back for handling mouse dragged events. Now the line is redrawn as you drag the mouse (while holding the left button down).

10.) (2 pts) **SHA1.** All of your work for this part of the lab should consist of modifications to the file `warmup.cpp` only. During grading, show how you compute the SHA1 of your `warmup.cpp`.