Assignment 2: Rotating a Cube in 3D Space

Given a set of data points that describe a cube centered at the origin in 3-dimensional space, our goal is to rotate the cube about an axis. This assignment will introduce Vertex Arrays in OpenGL for modeling sets of points in 3D space.

The skeleton code is provided to guide the assignment.

I have provided you with a simple degree to radian function for converting a given degree theta to radian as input to your rotation matrix. In addition, you are also given a `vector2array` function that converts a vector of `GLfloat` to an array of `GLfloat` to be rendered.

**Note: the C++ vector class is the equivalent to a list in most other languages.

You will complete the following functions for the assignment marked with `TODO`:
1) `to_homogenous_coord` : converts a vector of cartesian coordinates (x, y, z) to homogeneous coordinates (x, y, z, 1)
2) `to_cartesian_coord` : converts a vector of homogeneous coordinates (x, y, z, 1) to cartesian coordinates (x, y, z)
3) `rotation_matrix_x` : outputs the rotation matrix along the x-axis
4) `rotation_matrix_y` : outputs the rotation matrix along the y-axis
5) `rotation_matrix_z` : outputs the rotation matrix along the z-axis
6) `mat_mult` : performs matrix multiplication between two matrices

The camera has been set up for you to point towards the origin.

I have provided `points` (`vector<GLfloat>`) which contains the set of points defining the cube in 3D space. Your goal is to apply some rotation to the points in 3D space. I have also defined an array of GLfloat called `colors`. These colors are mapped to the planes so that you will be able to distinguish each plane of the cube.

You will notice that there is a global variable `theta`. `theta` defines the degree of rotation, which you will need to convert to radian using the provided `deg2rad` function.

Submission:

You will submit the following to Bright Space

1) "assignment2.cpp"
2) A recording (avi, mov) of the program running (should show a cube rotating about the axis or axes of your choice (e.g. rotate about x and y axes)

Grading:

I will be compiling the assignment using the following command:

```
gcc -o assignment2 assignment2.cpp -lGL -lGLU -lglut
```

Your code must compile for me to assign points!

Your assignment will be graded on:

1) the correctness of your implementation of the above functions

Late Policy:

For each day the assignment is late, 50% of its worth will be deducted, e.g. 100% on time, 50% 1 day late, 25% 2 days late, etc.