Section B

1.

1. Having a 4 x 4 matrices (homogenous) let us need only one set of operation to process all type of matrix operation that we are interested in, rather than doing complex transformation using the 3x3 matrix.
2. It is impossible to tell whether an object has moved or a camera. Draw a 2d diagram that has an object at point x = 2 and camera at point x = 4. Move the camera 2 moves to the right, the distance is 4, another one where you move the object 2 steps back and distance is also 4. Hence instead of moving an object we could move the camera.
3. 1- Construct matrix M1, which result from translating the vector A to pass by the origins.   
   2- Construct matrix M2, from rotating M1 about its X axis, mapping it into the XY -plane.  
   3- Construct matrix M3, from rotating M2 about its Z axis, mapping it into the X plane.  
   4- Perform the rotation by angle theta about the X axis on matrix M3, constructing matrix M4.  
   5- Reverse the effect of 1-3   
   A’ = M1-1 . M2-1 . M3-1 . M4 . M3 . M2. M1. A0
   1. Parallel projection: A projection is the process of mapping a 3D coordinates into 2D to view on the screen. Parallel projection is where set of points at which the projector (object) intersect the projection plane.
   2. Perspective projection: The items closer to view plane are bigger, while the one further away looks small, with edges and angles looking distorted. There are three types of perspective projections 1, 2, and 3 points which determined by respective vanishing points.
   3. Perspective division: We represent perspective using a 4x4 perspective projection matrix. If we multiply a point by a perspective transformation we are returned a point in the homogenous matrix. Perspective division return a homogenous matrix to a 3 point coordinate matrix and setting the fourth coordinate to 1.
   4. Near and far clip planes: The plane that define the view volume of a projection. Item in front of the near clip planes, items behind the far clip planes, are clipped out from the screen and as a result hidden.

----------

2.

1. Process of finding the screen pixels that intersect a polygon.   
   To efficiently scan-convert a triangle:  
   Start at top of the triangle, working down, scanline by scanline, fill the pixels in between each beginning and end of a scanline. This is a floating point algorithm and we have to keep rounding to nearest pixel grid.
2. During the scan conversion process, we generate a pixel P and we determine whether another project closer to the eye is mapping to P, if it is then we don’t update it, otherwise we do.  
   This made efficient with the help of the z-buffer, which is a data structure which store the depth z of a pixel p.  
   The process is like this:  
   Initialise each pixel to background.  
   Initialise each z-buffer value to max-depth  
   For each pixel P generated during the scan conversion  
   Check if z-cords of P < Z-buffer[P]  
   If it is :  
    Calculate the colours of P  
    Store the colour in P  
    Update the value of z-buffer[P]  
   Else   
    Do nothing and it won’t be visible, because there is something hiding it
3. Due to lack of precision leads to z-fighting, leading to incorrect rendering of pixels with similar z values. Can be minimised with glPolygonOffset() meaning each fragment's depth value will be offset after it is interpolated from the depth values of the appropriate vertices.
4. I = iaka, + ip / d’ [kd(N.L) + ks(V.R)N)]
   1. ambient illumination: is the general level of illumination caused by different reflection in the scene.   
      Ambient illumination = ia ka  
      Where ka is the ambient reflection coefficient, number between 0 and 1  
      ia is the intensity of the ambient light
   2. Diffuse reflection: is the absorption and general re-radiation of light, some lightwave are absorbed others are reflected, distance affects it.   
      Diffuse reflection = ip/d’ [kd (N.L)]  
      And d’ = kc + kld + kqd2  
      Where ip = original intensity at source   
      Kd is the diffuse reflection coefficient, number between 0 and 1  
      N is the surface normal  
      L is the direction of the light source, from the surface.  
      d = is the distance light travelled   
      Kc is the constant lighting distance coefficient   
      Kl is the linear lighting distance coefficient  
      Kq is the quadratic lighting distance coefficient
   3. Specular reflection: is the reflection at air-surface level, this reflects the colour of the surface taking into account the observer’s position.  
      Ispecular  = Iq \* Ks \* (R.V)N   
      Where   
      Ip = intensity of light at source  
      Ks = a constant coefficient between 0 and 1, and it is an approximation of inbound angle and wavelength.   
      R = reflected bound angle   
      V = the observer position relative to the surface.   
      Varying n results in the area affected by this
   4. Compute the average vertex normal between 3 vertices A, B, and C  
      Compute Pixel colours CA, CB, CC at each of those vertices  
      For each scanline  
      Average the colour CLeft between CA and CC  
      Average the colour CRight between CC and CB  
      Average the colour between CLeft and CRight along the scanline  
        
      Computing the colour is done using the local illumination model.

--------

Section C:

1. Thresholding: it is a simple method that replace the image pixel with black if it is over certain threshold (constant), or white if it is bellow. It aims to reduce misclassification. Finding the threshold is either manual where programmer tries to find best thresholding value that drive the best result. Or by usage of automated algorithms, such as Otsu method or iterative method.
2. - Iterative method: Start with threshold = 0, average all pixels left to the threshold, average all pixel right to threshold, then average the 2 averages, if it is equal to the threshold, return the threshold, otherwise increment threshold and repeat.   
   - Mode method: find 2 peaks in the histogram, and pick the minimum between them, that is the threshold. (But it doesn’t cope very well with a lot of contrast)  
   - P-tile: if you know the proportion of the image that is the object, threshold the image to select that proportion of the image (Relies a lot on knowing what exactly you’re looking for, and knowing exactly the data in your image, it is really doing this I see an image, oh I think value 140 would highlight my object, let me choose it. Pretty dumb and rarely works, but it is a method)  
   - Optimal classifier - fit 2 gaussian distributions to the histogram peaks, you’re looking to find a mean and variance for each such that you minimise misclassification.  
   - Manual: lulz just select w/e works to get this lab marked (actually a legit method yo)
3. the automated method would have get misguided by the parts that are brighter than the other parts and giving an overestimated threshold.  
   One way of dealing with it is by breaking the image into separate chunks, or even on a pixel level, but this will be expensive computationally.  
   If you’re using a perceptual colour space like hue saturation value then you can just ignore the value element (ie brightness)
4. First apply thresholding to provide a good contrast between the nerves and that blob thing in the background. Then we have two options:  
   1- Sharpen the image to highlight the edges in it, then we run it with Sobel edge detector, which detect the edge of the optic disk and highlight it by applying the convolution operation with a kernel that highlights left/top edges, and orientation, and combining that   
   2- (less likely) apply median filter to remove noise generated from thresholding to improve accuracy then sharpen the image to . Run a connected component labelling algorithm to be able to identify which components are connected (I am no medic but that circulus thing looks really really important). After that since it is the outline we are interested in a chain codes algorithm run alongside the outline to give us area, perimeter and whatever else information we need.   
   1. By applying an automatic method of finding a threshold, a method that works on minimising the invariance between class, such as Otsu’s method, or weighted histogram algorithm. After finding the threshold and threshold the image, I want the image to provide a good contrast between the nerves and that circulus thing behind the nerves that looks kinda important.
   2. There will be noise left from thresholding, I would run the image with a median filter to remove the noise, which works by finding a median value in a 3 x 3 matrix in the image. This would clean the noise and improve accuracy of classifying.
   3. First I would train it, and then test it against test cases, create a confusion matrix and do a cross validation on the data, ensure that we get a satisfactory % as correc.

----------------

4. FK ME :(