Question 1)

1. So we can do all matrix transformation operations we are interested using one operation rather than multiplying then adding.
2. Used for transforming the geometry drawn in a scene and specifically the camera. The main role is allowing us to transform the camera thus manipulating our view of the scene.
3. By doing series of matrix transformations, using what the OpenGL library provides such glTranslate, glRotate and so on.
4. I don’t think this is covered in our year.  
   However,  
   Matrix stack: allows the use of hierarchical models in OpenGL, allowing parent-child transformation to be composed easily.  
   E.g. The wheels of a car can be drawn without regard to parents location, with matrix stack it will be transformed alongside its parent.
5. 1- Create matrix M1, which translate vector V to pass through the origins.  
   2- Create matrix M2, by rotating M1 about its X axis mapping it into XY plane.  
   3- Create matrix M3, by rotating M2 about its Z axis mapping it into the X plane.  
   4- Perform your rotation about the X axis. Creating matrix M4.  
   5- reverse the effect of steps 1-3   
   Thus   
   V’ = M1-1.M2-1.M3-1.M4. M3. M2. M1. V

Question 2)

1. Modelling: Is the composition of the three dimensional scene, basically the skeleton of the scene.   
   Rendering: Processing of taking a scene and converting it to an image representation. This basically take care of lighting, interaction between objects, shading, etc…
2. 1. Ambient light: the general level of illumination caused by multiple reflection in the scene.   
      iambient = ia \* ka  
      Where ia is the ambient intensity (constant)  
      Ka is the ambient coefficient (value between 0 and 1, lower darker)
   2. Diffuse reflection: To model the effect of different angles of incidence and distance from light source.  
      Idiffuse = ipkdcos(theta) // or ipka(N.L)  
      Where  
      Ip intensity of the light source  
      Kd the diffuse coefficient   
      N is the surface normal  
      L is the direction of the light source from the surface  
      Theta is the angle of incidence   
        
      To model the effect of distance we do  
      Idiffuse = ip/d’ [kd(N.L)]  
      Where d’ = kc + kld + kqd2   
      Where kc  is the constant lighting coefficient   
      Kl is the linear lighting coefficient   
      Kq is the quadric lighting coefficient  
      D is the distance light has travelled
   3. Specular reflection: the reflection at air-surface interface.   
      Ireflection = ipks(R.V)n  
      Where  
      Ks is the approximated inbound angle, value between 0 and 1  
      R is the reflected outbound angle  
      V is the viewer’s position relative to the surface  
      N is a constant between 1 and 200  
        
        
      Putting all together we have the formula  
        
      i = iaka + ip/d’ [kd (N.L) + ks(R.V)n]
3. To introduce colour model we will need to represent each colour on its own.   
   For example for red colour it would be  
   Ir = iaRkaR + ipR/d’ [kdR(N.L) + ksR(R.V)n]   
   We will need to do same for blue and green, replacing R with B and G in previous formula.
4. Gouraud shading use interpolation to smooth out the discontinuities between polygons, it works by averaging the normals where polygon shades varies.  
   It first computes the average vertex normal at 3 points, A, B, C  
   Then compute the pixel colour Ca, Cb, Cc  
   For each scanline {  
    Average colour from Ca to Cc, calling it Cleft  
    Average colour from Cb to Cc, call it Cright  
    Average colour from Cleft to Cright along the scanline  
   }

Section C

Question 3.

1. Thresholding: is a technique to produce a binary image based on comparing a pixel of an image against a threshold (which we can either find automatically or set manually), if it is bigger than the threshold then it produce a black pixel, otherwise a white one (or different colours, but we use black and white to produce the maximum contrast which is usually the reason for thresholding)
2. P-Tile method: if you know the proportion of the image to be thresholded, then choose a threshold that represent the proportion you want highlighted (aka manually set up the threshold based on your knowledge of the image)  
   Mode method: Find 2 peaks in the histogram and take the lowest point in between them, this method doesn’t cope well with images with a lot of contrast in it.  
      
   Iterative method: Start with threshold = 0, average all pixel left of threshold, average all pixel right of threshold, then compute the average between them and compare it to threshold, if it is equal then we found the threshold, otherwise increment the threshold and do it again.  
   Otsu method: Work to minimise the intra-class variance.
3. The automated method would get misguided by the contrast and choose an incorrect threshold (one that lose too many information)  
   One way of solving it is to breakdown the image pixels into sections (or even pixels) and thresholding each, but this is very expensive computationally.
4. First capture the image.  
   Then we have two choices:  
     
   Preferably to use the first, but depend on what we need in the future more we might want to use the second.  
   First algorithm is relying on colour distribution, second algorithm relying on detecting the seals themselves.   
   First algorithm: Apply median filtering on the image to remove the noise created by the camera and smooth out the image. Then apply connected component analysis, using the result we will apply the colour distribution algorithm that will detect the dark seal from the white background. This can easily estimate and compute the number of seals.   
     
   Second: Since there is already a heavy contrast, we can threshold the image to produce an even bigger contrast, this will produce a binary image where the seals will be completely black and ice/snow will be completely white.  
   Then applying the median filter to smooth out the image and remove the noise introduced from the camera and thresholding.  
   Then now we have blobs in our image, seals will always look the same, so we do a connected component analysis on the seals.   
   Now we need to train our algorithm using central moment of area to get the different shapes and orientation seals can be at.   
   Then we can test it on different seals and include non seals in our tests, produce a confusion matrix and do cross validation to ensure our algorithm is working correctly.

Question 4.

SCREW ME!