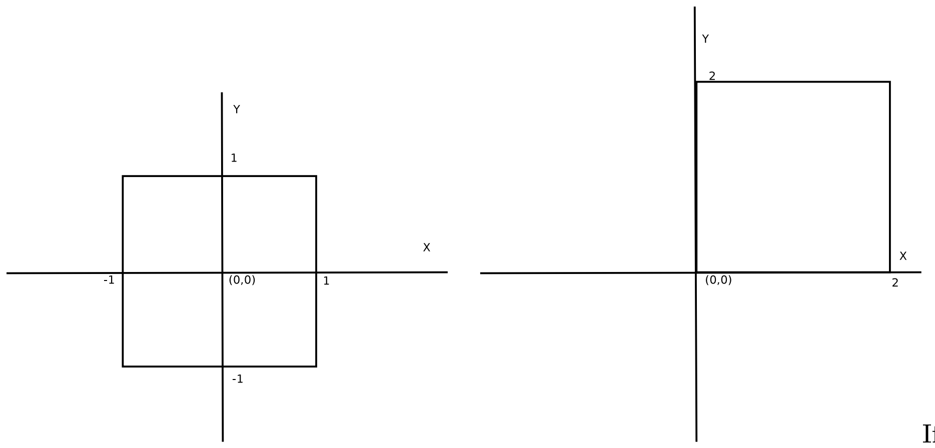
2 Sequence of transformations

The original and final configurations are shown in the picture. 3 points for just drawing the figure with the right coordinates. No partial credit.



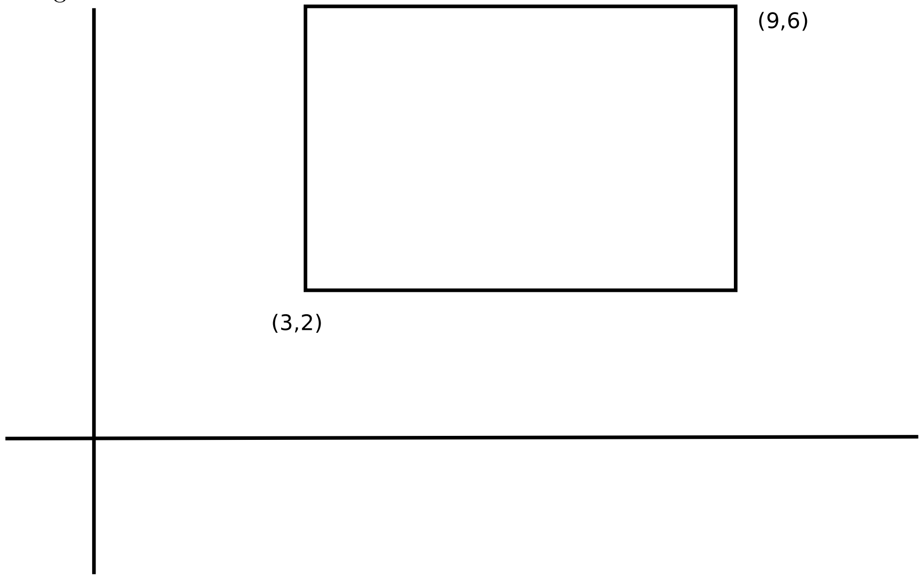
If they say that the same transformation can be achieved by a glTranslatef(1, 1, 0), they get 2 points. But realize that this, although looks the same in this drawing, displays a 90 degrees rotated version of the square. A more correct answer (for full 4 points) would be:

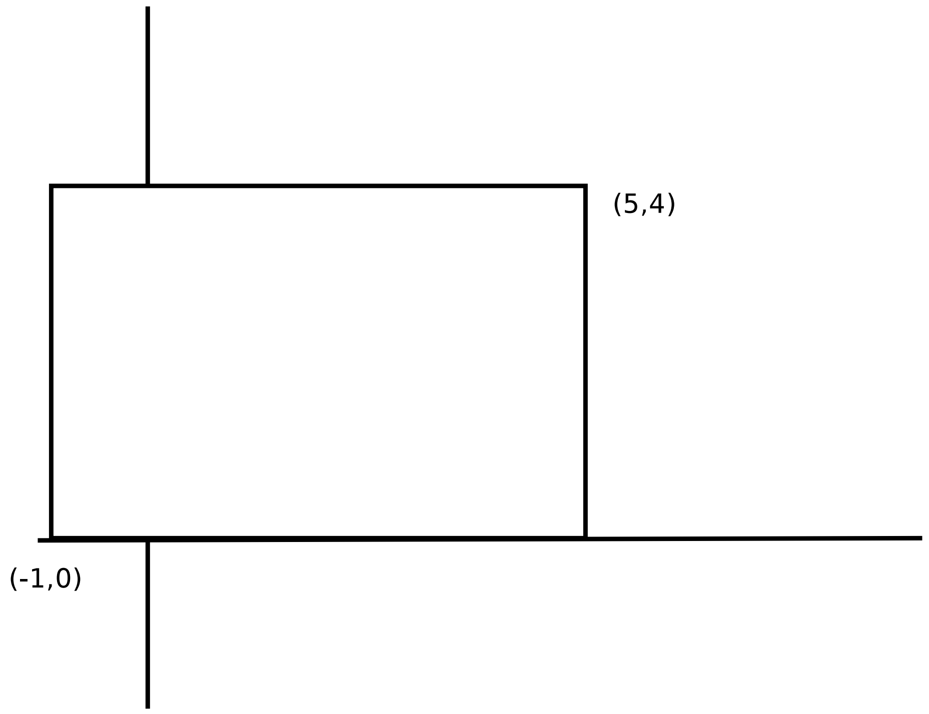
- glTranslatef(1, 1, 0)
- glRotatef(90, 0, 0, 1)

The same transformations in a different order would get 1 point less.

3 More transformations

The images resulting from case 1 and 2 (3 points each, no partial credit) are the following:





To make case 2 be the same as case 1, it should be (4 points, no partial credit):

- glTranslatef(6,4,0);
- glScalef(3,2,0);

The analysis required is that the translation parameters have to be scaled (multiplied) by the scaling parameters, because in the first case, the scaling matrix is applied before the translation, and the translation gets scaled. This change in the parameters corrects for that. (2 points for the right answer, or 1 for a partial but correct observation).

4 Inverse of rotation matrices (10 pts.)

Row vectors are unit long and perpendicular among themselves (orthonormal), and so are column vectors. In a rotation matrix, this shows up in the fact that the dot product of a row with itself is equal to 1, but the dot product of a row with another row is 0. The product of a rotation matrix R and its inverse (transpose) R^T is the identity matrix. That's natural if we realize that the position (i, j) of the matrix product $M \cdot N$ is given by the dot product of the row i of matrix M with column j of matrix N.

[This question is hard to split in parts, so I'll give it to you to give partial credit, depending on how close to the solution they are].

6 Commutativity of rotation matrices

They have to prove this by laying out the rotation matrices for both rotations, and seeing how the expressions reduce to the same. For example, for the first one, $R_z(\theta_1) \cdot R_z(\theta_2) = \begin{pmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} =$ $\begin{pmatrix} c_1c_2 - s_1s_2 & -c_1s_2 - s_1c_2 & 0 \\ c_1s_2 + s_1c_2 & c_1c_2 - s_1s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. And if the order of the multiplied matrices is the opposite, the result should be the same. This implies that rotations about the same coordinate axis are commutative. [This part receives 4 points.]

The second part is simpler, and just requires some high school trigonometry, like sin(a+b)=sin(a)cos(b)+cos(a)sin(b), and cos(a+b)=cos(a)cos(b)-sin(a)sin(b). [Just 2 points in this part].

Finally, in the third part (worth 4 points), they have to decompose a rotation matrix into simpler ones, like this: $R(\theta_1) = R_x(-\alpha_1)\, R_y(-\lambda_1)\, R_z(\theta_1)\, R_y(\lambda_1)\, R_x(\alpha_1)$. So when we put two of these expressions together, the result is easily simplified:
$R(\theta_1)\, R(\theta_2) = R_x(-\alpha_1)\, R_y(-\lambda_1)\, R_z(\theta_1)\, R_y(\lambda_1)\, R_x(\alpha_1)\, R_x(-\alpha_1)\, R_y(-\lambda_1)\, R_z(\theta_2)\, R_y(\lambda_1)\, R_x(\alpha_1) =$
$= R_x(-\alpha_1)\, R_y(-\lambda_1)\, R_z(\theta_1)\, R_y(\lambda_1)\, R_y(-\lambda_1)\, R_z(\theta_2)\, R_y(\lambda_1)\, R_x(\alpha_1) =$
$= R_x(-\alpha_1)\, R_y(-\lambda_1)\, R_z(\theta_1)\, R_z(\theta_2)\, R_y(\lambda_1)\, R_x(\alpha_1) =$
$= R_x(-\alpha_1)\, R_y(-\lambda_1)\, R_z(\theta_1 + \theta_2)\, R_y(\lambda_1)\, R_x(\alpha_1) = R(\theta_1 + \theta_2)$.

7 Scaling along a particular vector

This should be done by composing a translation by $(-5, -5, -5)$, a rotation, a scaling by a factor of 3, then the inverse of the rotation and the inverse of the translation. The rotation matrix R should be such that aligns the vector $(1, 2, 1)$ with a coordinate axis (for example, Z). You can tell because vector $(1, 2, 1)$ should appear in one of the rows. The scaling matrix S should be an identity matrix with a 3 in the position of the diagonal which is aligned with the row of vector $(1, 2, 1)$ in the rotation matrix. So if the student chose to align vector $(1, 2, 1)$ with axis Z, then the diagonal of the scaling matrix should have the third component as a 3, and all others as 1.

The result should look like this: $T(5, 5, 5)\,R^T SRT(-5, -5, -5)$.

If they don't give the intermediate steps but just the final matrix, check that it's right as follows: Post-multiply vector $(5, 5, 5)$ by their matrix, and the result should be $(5, 5, 5)$. Then post-multiply $(6, 7, 6)$, and the result should be $(8, 11, 8)$; finally, post-multiply $(1, 5, 1)$, and the result should remain the same.

It is OK if they don't give the final matrix, but just the explanation and the decomposition into 5 matrices. [The explanation and the correct result is worth 10 points; If the result is wrong, but the explanation fine, give them 8 points; If the result is fine but the explanation is missing or wrong, give them no more than 3 points].

8 gluLookAt and glFrustum

The first task is to find the "center" position, which is the projection of the eye point $(0, 0, 0)$ onto the image plane. The unnormalized plane normal is $(1, 1, 1)$. It's easy to infer that the projection of $(0, 0, 0)$ onto the plane has the form $(0, 0, 0) + k(1, 1, 1)$, and it's exactly $(2, 2, 2)$. As expected, plugging these coordinates into the equation makes it hold true $2 + 2 + 2 = 6$. So now we know that the near clipping plane is at distance $\sqrt{2^2 + 2^2 + 2^2} = \sqrt{12} = 2\sqrt{3}$. This will be useful for the second part of the question. Now we have the eye point $(0, 0, 0)$, the center point $(2, 2, 2)$, and the up vector $(0, 2, 0)$, which normalized is $(0, 1, 0)$.

The calculation process for gluLookAt is well explained here:
(http://pyopengl.sourceforge.net/documentation/manual/gluLookAt.3G.html), although there is a missing normalization in the calculation of s.

We let f be the normalized vector (center-eye), or $f = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right)$. $s = \frac{f \times up}{|f \times up|}$, and $u = s \times f$. We construct matrix $M = \begin{pmatrix} s0 & s1 & s2 & 0 \\ u0 & u1 & u2 & 0 \\ -f0 & -f1 & -f2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, which has the form: $M = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} & 0 \\ -0.40825 & -0.8165 & -0.40825 & 0 \\ \frac{-1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. [Getting the complete development gets 6 full points, even if the matrix is not the same.]

For the glFrustum, we know n, f, l, r, t and b, which is all we need to compute the matrix as explained in:
(http://www.talisman.org/opengl-1.1/Reference/glFrustum.html).

So we have:
$A = (right + left)/(right - left)$
$B = (top + bottom)/(top - bottom)$

$C = -(far + near)/(far - near)$
$D = -(2 * far * near)/(far - near)$
$E = (2 * near)/(right - left)$
$F = (2 * near)/(top - bottom)$

And the matrix is $P = \begin{pmatrix} E & 0 & A & 0 \\ 0 & F & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1.73205 & 0 & 0 & 0 \\ 0 & -1.73205 & -3 & 0 \\ 0 & 0 & -2.06002 & -10.60023 \\ 0 & 0 & -1 & 0 \end{pmatrix}$.

[Again, 6 points for the development, independently of the result.]

Finally, in order to compute the projected coordinates of $p = (10, 4, 6)^T$, we simply multiply $p_{pr} = P \cdot M \cdot p$, and the result should be $(-4.8990, 40.2979, 13.1869, 12.5470)^T = (-0.39045, 3.21175, 1.051, 1)^T$. [The correct result, or approximately correct result gets 2 points; The correct procedure, but wrong result gets 1 point. Partially correct procedure (missing normalizations, etc) get points off according to the importance of the mistake.]

3 Model transformation

The transformation is a rotation $R = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ followed by a translation $T = \begin{pmatrix} 1 & 0 & 0 & 20 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

3 points for each matrix.

4 Transformation order

The answer is B: TRv.

No partial credit.

5 Identity matrix

It means that the camera is at the origin, looking towards the negative Z axis, with the up vector along the positive Y axis.