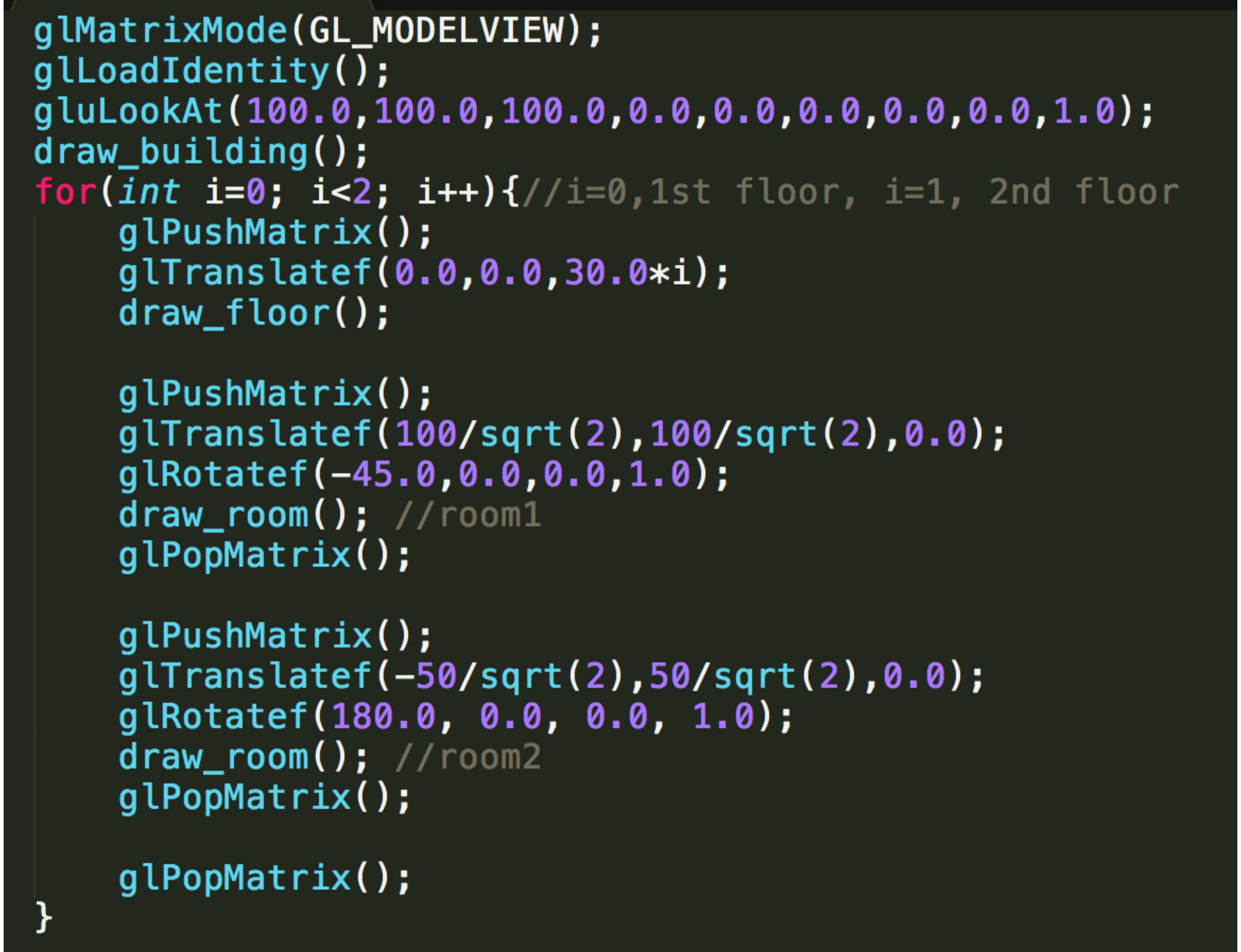
Question1&2 We are looking for…

- Push pop order (15’)
- Translation rotation order (8’) – multiple solutions
- f1 f2 r1 r2 in numerical order (2’)
- Value in glTranslate and glRotate (4’)
- 2 floors and 4 rooms are drawn (6’)

Q1

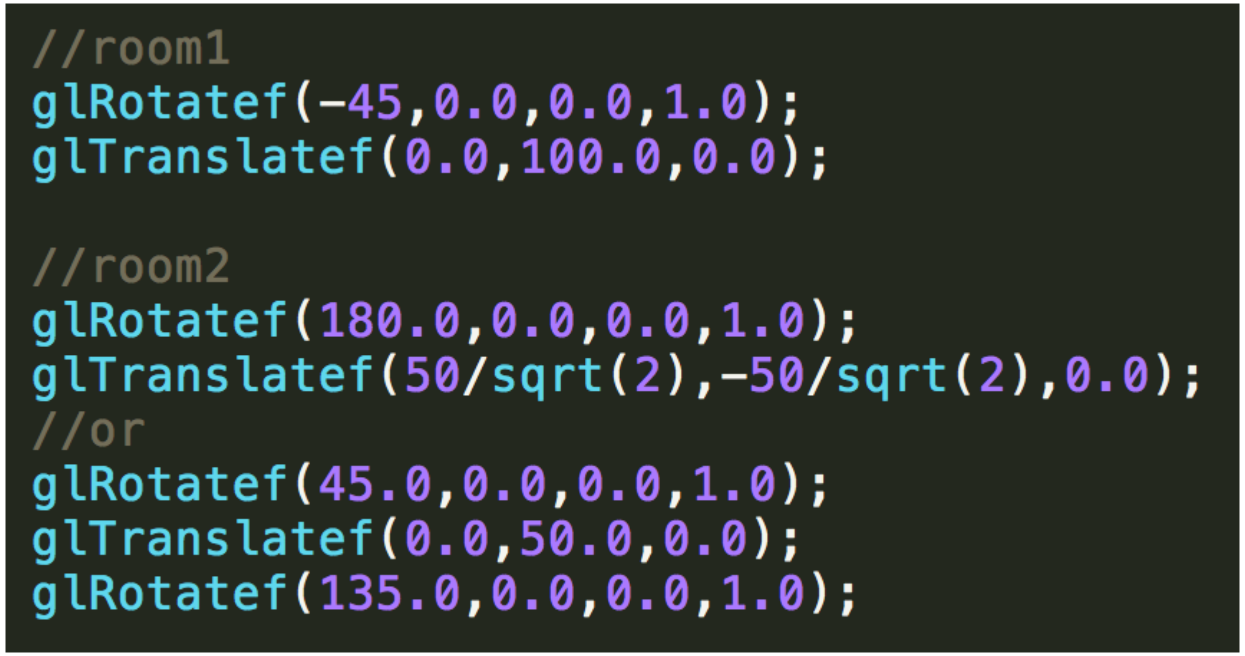
not using loops are ok



```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(100.0,100.0,100.0,0.0,0.0,0.0,0.0,0.0,1.0);
draw_building();
for(int i=0; i<2; i++){//i=0,1st floor, i=1, 2nd floor
    glPushMatrix();
    glTranslatef(0.0,0.0,30.0*i);
    draw_floor();

    glPushMatrix();
    glTranslatef(100/sqrt(2),100/sqrt(2),0.0);
    glRotatef(-45.0,0.0,0.0,1.0);
    draw_room(); //room1
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-50/sqrt(2),50/sqrt(2),0.0);
    glRotatef(180.0, 0.0, 0.0, 1.0);
    draw_room(); //room2
    glPopMatrix();

    glPopMatrix();
}
```

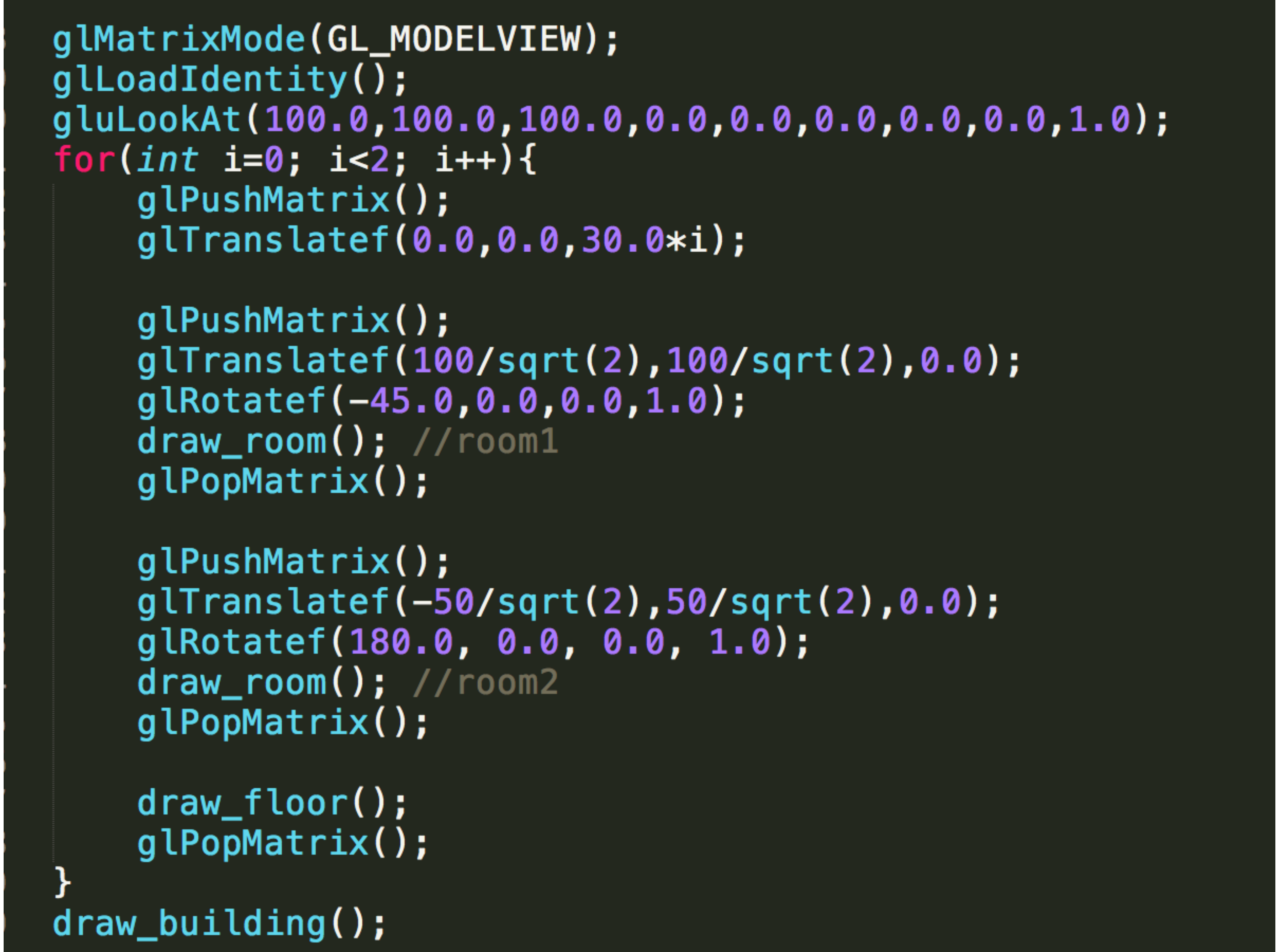
If you want to view TR as coordinate change or want to write in the R T order

- These are also accepted



```
//room1
glRotatef(-45,0.0,0.0,1.0);
glTranslatef(0.0,100.0,0.0);

//room2
glRotatef(180.0,0.0,0.0,1.0);
glTranslatef(50/sqrt(2),-50/sqrt(2),0.0);
//or
glRotatef(45.0,0.0,0.0,1.0);
glTranslatef(0.0,50.0,0.0);
glRotatef(135.0,0.0,0.0,1.0);
```

Q2



```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(100.0,100.0,100.0,0.0,0.0,0.0,0.0,0.0,1.0);
for(int i=0; i<2; i++){
    glPushMatrix();
    glTranslatef(0.0,0.0,30.0*i);

    glPushMatrix();
    glTranslatef(100/sqrt(2),100/sqrt(2),0.0);
    glRotatef(-45.0,0.0,0.0,1.0);
    draw_room(); //room1
    glPopMatrix();

    glPushMatrix();
    glTranslatef(-50/sqrt(2),50/sqrt(2),0.0);
    glRotatef(180.0, 0.0, 0.0, 1.0);
    draw_room(); //room2
    glPopMatrix();

    draw_floor();
    glPopMatrix();
}
draw_building();
```

Q3

- a. Explain functions (5’)
  Preserves (5’)
  Reason (5’)

The code first set the viewport (how many pixels you want to use in the new window) to be w and h, starting from (0,0), namely the program will use the full window to display.
Then the code rest the projection matrix with aspect ratio w/h, and corresponding field of view (fov), near and far planes.
Finally it switch back to the MODELVIEW mode for other manipulations.

It can preserve the object’s aspect ratio shown in the scene. Since it change the projection matrix to operate on an image plane with changed aspect ratio, the image plane for this new matrix corresponds to the new viewport size.

- b. glViewport(0,0,min(w,h),min(w,h)); (10’)
  …
  gluPerspective(80.0, 1.0, 1.0, 50.0); (5’)

REMINDER

<u>In the second assignment, you need to:</u>

1. Animate a block by rotating it about the vertical axis passing through the center of the block. You should rotate some blocks clockwise and other counterclockwise. All five blocks must be in motion. Again, fancier motion involving, say rotation, translation and scaling are allowed.
2. Animate the string of numbers (from 0 to 9) into the scene. The numbers must be opaque and colored. You can design your own animation trajectory.
3. Apply a wood texture to all faces of all blocks.
4. Turn on the light (there should be a single light source on top of the blocks). Use an appropriate OpenGL lighting model to shade the scene.
5. Add the ability to change the viewpoint. The user can move the mouse up and down to move the camera up or down, and move the mouse left and right to move the camera left and right. In any case, the camera's look-at point should be in the center of the scene. The easiest way to implement change of view point is through “press-and-drag.” That is, the user presses down on the left mouse button and drags the mouse to a new location. You should interactively update the viewpoint based on some function of the cursor displacement until the mouse button is released.
6. Add the ability to have shadow. Only “fake” shadow is needed (e.g., shadow cast on a big background plane).