Assignment Problem:

Problem: Use texture mapping, to wrap numbers around a cube that rotate itself. Write code to transfer the UV data to the OpenGL buffer. You will then write the code to bind the texture in the loop and shader code to draw the texture.

Methods: With the code given, the projection matrix and most of the GL loop code are already written for you, which is similar to the last assignment and methods without the camera this time. The calculation of the vs file will be gl_position = projection * view * model * vec4(position, 1.0) in that order just like the previous assignment which will create the cube that the texture will wrap around. The UV will also need to be assigned to the respective matrices and layouts, which uses the code: UV = vertexUV.

The frag file will use the sampler2d variable for the sampler using the texture function, and since the cube needs to be wrapped all around the cube, the second parameter will be vec2(UV.x, 1.0 - UV.y) where the y has to be inverted in order for all the numbers to show up as the coordinates in the actual texture file are grabbed diagonally. The command will be color = texture(myTextureSampler, vec2(UV.x, 1.0 - UV.y)).

Next, you have to set up the UV buffer in mian.cpp which creates the buffer that loads the coordinates of where to place the texture around the cube and how to wrap it, since it is a 3D object. You create the buffer, bind it, then add data to the UVBO with the respective commands. glGenBuffers(1, &UVBO) with parameters of size 1, and the address to the UVBO int variable which holds the data for the VBO, next step is glBindBuffer(GL_ARRAY_BUFFER, UVBO) which will bind the buffer to the array and bind the UVBO data to it. This is so that the other GL functions can use the buffer on demand. Next you add the data via glBufferData(GL_ARRAY_BUFFER, sizeof(uv), uv, GL_STATIC_DRAW); which generates the data object and initializes it.

Last step will be to bind the buffer in the game loop which will look for the object and bind the texture to it each frame. First thing is to enable the array to edit, specifically array 1 using the command glEnableVertexAttribArray(1); Next you bind the buffer to the UVBO which will allow you to use the buffer and alter the data on it, which would be glBindBuffer(GL_ARRAY_BUFFER, UVBO) which takes the arguments of buffer array and your glint variable which holds the data. Finally, glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE, 0, (void *)0); tells the rendering software how to interpret the array data and can alter based on the size argument (the second one) that is given to it.

Results: The result is a number on each side of the cube that rotates automatically. The results are also posted into 2 screenshots that are accessible in the github directory. The orientation depends on the numbers calculated in the vs file and messing with them will give various results.