The goal of this homework assignment was to create several SO(3) rotation matrices for a cube, then create a program that will randomly multiply said matrices until all the elements of their group are found. All computations were done in Python using Numpy. The output of the code gave the following results:

1. The identity matrix:
$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

2. Rotations about axes (90°):
Z axis:
$$90 = \begin{bmatrix}0 & -1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{bmatrix} \quad 180 = \begin{bmatrix}-1 & 0 & 0\\0 & -1 & 0\\0 & 0 & 1\end{bmatrix} \quad 270 = \begin{bmatrix}0 & 1 & 0\\-1 & 0 & 0\\0 & 0 & 1\end{bmatrix}$$
Y axis:
$$90 = \begin{bmatrix}0 & 0 & -1\\0 & 1 & 0\\1 & 0 & 0\end{bmatrix} \quad 180 = \begin{bmatrix}-1 & 0 & 0\\0 & 1 & 0\\0 & 0 & -1\end{bmatrix} \quad 270 = \begin{bmatrix}0 & 0 & 1\\0 & 1 & 0\\-1 & 0 & 0\end{bmatrix}$$
X axis:
$$90 = \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & -1 & 0\end{bmatrix} \quad 180 = \begin{bmatrix}1 & 0 & 0\\0 & -1 & 0\\0 & 0 & 1\end{bmatrix} \quad 270 = \begin{bmatrix}1 & 0 & 0\\0 & 0 & -1\\0 & 1 & 0\end{bmatrix}$$

3. Rotations about corners (120°):
Four corners can be rotated about: 1, 2, 3, 4.

   (a) Corner 1
$$120 = \begin{bmatrix}0 & -1 & 0\\0 & 0 & -1\\1 & 0 & 0\end{bmatrix} \quad 240 = \begin{bmatrix}0 & 0 & 1\\-1 & 0 & 0\\0 & -1 & 0\end{bmatrix}$$
   (b) Corner 2
$$120 = \begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\1 & 0 & 0\end{bmatrix} \quad 240 = \begin{bmatrix}0 & 0 & 1\\1 & 0 & 0\\0 & 1 & 0\end{bmatrix}$$
   (c) Corner 3
$$120 = \begin{bmatrix}0 & -1 & 0\\0 & 0 & 1\\-1 & 0 & 0\end{bmatrix} \quad 240 = \begin{bmatrix}0 & 0 & -1\\-1 & 0 & 0\\0 & 1 & 0\end{bmatrix}$$

(d) Corner 4

$$120 = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \quad 240 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

4. Edge rotations (180°):

$$1,2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad 1,3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \quad 2,4 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

$$3,4 = \begin{bmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad 1,5 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \end{bmatrix} \quad 2,6 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The program, after computing all of these matrices, then computed the determinante of each one to check if they are all in SO(3). After this, it counted all of the matrices and printed them out. In addition, the code help a matrix representing the cube, which was used for determining which matrices corresponded to which rotations. **The group of rotations consists of 24 unique matrices, all of which are SO(3).**

For the bonus, the following new rotation was introduced to the group that was not computed initially in order to create a new, larger group:

$$\begin{bmatrix} \cos(175°) & -\sin(175°) & 0 \\ \sin(175°) & \cos(175°) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

After doing several trials, one with 10,000 iterations, one with 20,000 iterations, and one with a 1,000,000 iterations, it was concluded that a new group was not formed, or if one was, it was an infinite group. The first trial created 9917 matrices, while the second created 19960 matrices. The third trial was too taxing for the student's computer, so was prematurely ended. Another trial was conducted with a new angle of 170°, which yielded similar results.

Appendix

Cube Rotations Code

```
#!/usr/bin/python
import numpy as np

#Create a cube centered around the origin
cube = np.matrix(((-1,1,-1,-1,1,-1,1,1),(-1,-1,1,-1,-1,1,1,1),(-1,-1,-1,1,1,1,-1,1)))

################################################################################################
"""
Defining the matrices initially used to find all rotations.
"""

#The Identity matrix
identity = np.matrix(((1,0,0),(0,1,0),(0,0,1)))

#90 degree rotation about z axis
zrot = np.matrix(((0,-1,0),(1,0,0),(0,0,1)))

#180 degree rotation
rot180 = np.matrix(np.array([(0,0,1),(0,-1,0),(1,0,0)]))

#120 degree rotation
rot120 = np.matrix(np.array([((0,1,0),(0,0,1),(1,0,0))]))

################################################################################################
"""
Compiling more matrices.
"""

rotations = [identity,zrot,rot180,rot120]

"""
Stole from Hollis :)
"""

for i in range(10000):
    count = 0
    random1 = np.random.randint(len(rotations))
    random2 = np.random.randint(len(rotations))
    newrot = rotations[random1]*rotations[random2]
    for j in range(len(rotations)):
```

```
        if np.all(newrot == rotations[j]):
            continue
        else:
            count += 1
    if count == len(rotations):
        print("New matrix added!")
        rotations.append(newrot)

#print("There are " + str(len(rotations)) + " matrices to rotate the cube.")

###############################################################################
"""
Making sure all matrices are SO(3).
"""

SO3check = 0

for i in rotations:
    if np.linalg.det(i) == -1:
        SO3check += 1

if SO3check == 0:
    print("All matrices are SO(3)!")

else:
    print("Some matrices are not SO(3)!")

###############################################################################

"""
Prints each matrix.
"""

for rot in rotations:
    print(rot)
    print('--------------------')

###############################################################################

"""
Bonus with new angle rotation!
"""

rotationplus = []
for rot in rotations:
    rotationplus.append(rot)
```

```
angle = np.radians(175)
cos = np.cos(angle)
sin = np.sin(angle)

rot175 = np.matrix(((cos,-sin,0),(sin,cos,0),(0,0,1)))
rotationplus.append(rot175)
for rot in rotationplus:
    print(rot)
    print('----------------')

for i in range(1000000):
    count = 0
    random3 = np.random.randint(len(rotationplus))
    random4 = np.random.randint(len(rotationplus))
    newrot = rotationplus[random3]*rotationplus[random4]
    for j in range(len(rotationplus)):
        if np.all(newrot == rotationplus[j]):
            continue
        else:
            count += 1
    if count == len(rotationplus):
        print("New matrix added!")
        rotationplus.append(newrot)

print("There are " + str(len(rotationplus)) + " matrices to rotate the cubeplus.")
################################################################################
```