Problem 1

1.1

```
##### STUDENT CODE START #####

Pw = np.zeros([4, 3])

a = [-s/2, -s/2, 0]
b = [s/2, -s/2, 0]
c = [s/2, s/2, 0]
d = [-s/2, s/2, 0]

Pw[0] = a;
Pw[1] = b;
Pw[2] = c;
Pw[3] = d;


##### STUDENT CODE END #####

return Pw
```

In this question, we are asked to calculate world coordinates for a,b,c,d. Since we know the length of the April tag, we can simply divide that by 2 and assign relevant +ve and -ve signs to each point to form the 4 corners of the tag.

Problem 2

2.1

```
# step 1: recover H from "est_homography"
world_coord = np.zeros([4,2])
world_coord[:,0] = Pw[:,0]
world_coord[:,1] = Pw[:,1]
```

```
# camera ~ H * world    Y ~ H * X
H = est_homography(world_coord, Pc)
```

The first step is to recover the H matrix from the "est_homography" file given. Solving "Camera ~ H * World" gives the H to transform from world coordinate to camera coordinate.

```
# step 2: K^-1 H = (R t) = (h1' h2' h3')
RT_matrix = np.zeros([3,3])
RT_matrix = np.matmul(np.linalg.inv(K), H)

h1_ = RT_matrix[:,0]
h2_ = RT_matrix[:,1]
h3_ = RT_matrix[:,2]

A = np.zeros([3,3])
A[:,0] = h1_
A[:,1] = h2_
A[:,2] = np.cross(h1_, h2_)
```

Next, we use the equation "K^-1 H = (R t) = (h1' h2' h3') to find h1', h2' and h3' which will be use in the next step.

```
A = np.zeros([3,3])
A[:,0] = h1_
A[:,1] = h2_
A[:,2] = np.cross(h1_, h2_)
# step 3: use SVD to find R and T
[U, S, V] = np.linalg.svd(A)
diagonal_matrix = np.eye(3)
diagonal_matrix[2,2] = np.linalg.det(U @ V)
```

Next, we can use SVD of A to find the rotation matrix R.

```
R = U @ diagonal_matrix @ V

R = np.transpose(R)

t = h3_ / np.linalg.norm(h1_)

t = -R@t
```

Lastly, we can use equation t = h3'/||h1'|| to find t. After that, Rwc and twc can be found by taking transpose of R and _Rt.

Problem 3

3.2

```
# step 1: set a, b, c
P1, P2, P3 = Pw[0], Pw[1], Pw[2]
P1c, P2c, P3c = Pc[0], Pc[1], Pc[2]

a = np.linalg.norm(P2 - P3)
b = np.linalg.norm(P1 - P3)
c = np.linalg.norm(P1 - P2)
```

First, choose 3 points and calculate a, b and c.

```
# step 2: find unit vectors j1 (FOR P1), j2 (FOR P2), j3 (FOR P3)
focal = [K[0, 0], K[1, 1]]
f = (focal[0] + focal[1]) / 2
center = [K[0, 2], K[1, 2]]
u0, v0 = center[0], center[1]

u1, v1 = P1c[0], P1c[1]
u2, v2 = P2c[0], P2c[1]
u3, v3 = P3c[0], P3c[1]
u1 = u1 - u0
u2 = u2 - u0
u3 = u3 - u0
v1 = v1 - v0
v2 = v2 - v0
v3 = v3 - v0
j1_vector = np.array([u1, v1, f])
j2_vector = np.array([u2, v2, f])
j3_vector = np.array([u3, v3, f])
j1 = j1_vector / np.linalg.norm(j1_vector)
j2 = j2_vector / np.linalg.norm(j2_vector)
j3 = j3_vector / np.linalg.norm(j3_vector)
#print(np.linalg.norm(j1), np.linalg.norm(j2), np.linalg.norm(j3))
# step 3: find alpha, beta, gamma from j1, j2, j3
alpha = np.arccos(np.dot(j2, j3))
beta = np.arccos(np.dot(j1, j3))
gamma = np.arccos(np.dot(j1, j2))
#print(alpha, beta, gamma)
```

Next, take in u0,v0 and f. Calculate unit vectors j1, j2 and j3.

```python
# step 4: enter coefficients A0, A1, A2, A3, A4
A4 = ((a**2 - c**2)/b**2 - 1)**2 - ((4*c**2)/ b**2) *
    (math.cos(alpha))**2
A3 = 4 * (((a**2 - c**2)/b**2) * (1 - (a**2 - c**2)/b**2) *
    math.cos(beta) - (1 - (a**2 + c**2)/b**2)* math.cos(alpha) *
    math.cos(gamma) +
    2*(c**2/b**2)*(math.cos(alpha))**2*math.cos(beta))
A2 = 2*(((a**2 - c**2)/b**2)**2 - 1 + 2*((a**2 -
    c**2)/b**2)**2*(math.cos(beta))**2 + 2*((b**2 - c**2)/b**2)*
    (math.cos(alpha))**2-4*((a**2 +
    c**2)/b**2)*math.cos(alpha)*math.cos(beta)*math.cos(gamma) +
    2*((b**2 - a**2)/b**2)*(math.cos(gamma))**2)
A1 = 4 *(-((a**2 - c**2)/b**2)*(1+(a**2 -
    c**2)/b**2)*math.cos(beta) +
    2*(a**2/b**2)*(math.cos(gamma))**2*math.cos(beta) - (1-(a**2 +
    c**2)/b**2)*math.cos(alpha)*math.cos(gamma))
A0 = (1+(a**2 - c**2)/b**2)**2 - 4*(a**2/b**2)*(math.cos(gamma))**2

AA = np.array([A4, A3, A2, A1, A0])
roots = np.roots(AA)
#print(roots[0], roots[3])
v_1, v_2 = roots[0].real, roots[3].real
```

Type in the A coordinates and use np.roots to get 4 roots. Take out those 2 positive ones to compute for u.

```python
u_1 = ((-1+(a**2 - c**2)/b**2)*v_1**2-2*((a**2 -
    c**2)/b**2)*math.cos(beta)*v_1+1+((a**2 -
    c**2)/b**2))/(2*(math.cos(gamma)-v_1*math.cos(alpha)))
```

Second u does not give good result, so I stick to u1 and v1.

```python
u_, v_ = u_1, v_1
s1 = (c**2/(1+u_**2-2*u_*math.cos(gamma)))**0.5
#print("s1", s1)
s2 = u_ * s1
s3 = v_ * s1
# calculate 3D coordinates in Camera frame
p1c = s1 * j1
p2c = s2 * j2
p3c = s3 * j3
pc = np.zeros([3,3])
pc[0] = p1c
pc[1] = p2c
pc[2] = p3c
```

Calculate s1, s2, s3 and p1c, p2c, p3c. R, t = Procrustes(pc, pw) will give R and t.

```
A = Y
B = X

A_ = (A[0] + A[1] + A[2]) / 3
B_ = (B[0] + B[1] + B[2]) / 3

A = np.array([A[0] - A_, A[1] - A_, A[2] - A_]).transpose()
B = np.array([B[0] - B_, B[1] - B_, B[2] - B_]).transpose()
```

Inside function Procrustes(), assign Y to A, X to B. Calculate the centroid for A and B. Form new matrix and transpose them to be use in the next step.

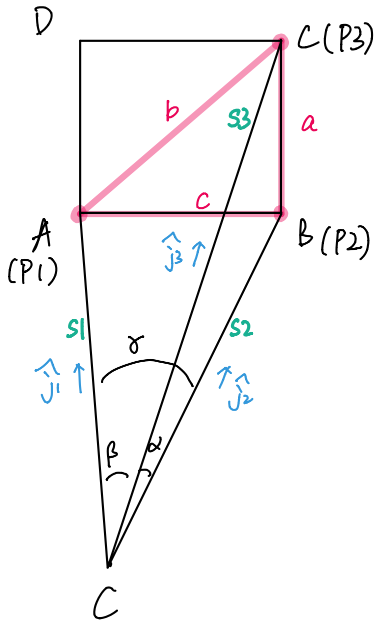
```
[U, S, V] = np.linalg.svd(B @ np.transpose(A))
diagonal_matrix = np.eye(3)
diagonal_matrix[2, 2] = np.linalg.det(np.transpose(V) @
    np.transpose(U))
R = np.transpose(V) @ diagonal_matrix @ np.transpose(U)
```

Follow lecture slides, SVD(BA^T) leads to rotation R, as stated in the code.

```
t = A_ - R @ B_
```

Finally, we get the transformation t.

Problem 3.1:



Step 1: find a, b, c

a, b, c are lengths. They can be found from taking the norm of a, b, c in world coord.

$a = \|p_2 - p_3\|$

$b = \|p_1 - p_3\|$

$c = \|p_1 - p_2\|$.

Step 2: find unit vectors $\hat{j}_1$, $\hat{j}_2$, $\hat{j}_3$.

P_c gives a, b, c, d in pixels (u_i, v_i)

$$\hat{j}_i = \left\| \begin{pmatrix} u_i - u_0 \\ v_i - v_0 \\ f \end{pmatrix} \right\| ,$$

where u_0, v_0 is given in K; f is the average of f_x, f_y focal length.

Step 3: find α, β, γ.

From dot product:

$\alpha = \cos^{-1}(\hat{j}_2 \cdot \hat{j}_3)$

$\beta = \cos^{-1}(\hat{j}_1 \cdot \hat{j}_3)$

$\gamma = \cos^{-1}(\hat{j}_1 \cdot \hat{j}_2)$.

Step 4: find S_1, S_2, S_3, from Grunert's Solution.

Solve eqn: $A_4 v^4 + A_3 v^3 + A_2 v^2 + A_1 v + A_0 = 0$, where A_4, A_3, A_2, A_1, A_0 are as follow:

$$A_4 = \left(\frac{a^2 - c^2}{b^2} - 1\right)^2 - \frac{4c^2}{b^2}\cos^2\alpha$$

$$\begin{aligned} A_3 = 4\Bigg[& \frac{a^2 - c^2}{b^2}\left(1 - \frac{a^2 - c^2}{b^2}\right)\cos\beta \\ & - \left(1 - \frac{a^2 + c^2}{b^2}\right)\cos\alpha\cos\gamma \\ & + 2\frac{c^2}{b^2}\cos^2\alpha\cos\beta\Bigg] \end{aligned}$$

$$\begin{aligned} A_2 = 2\Bigg[& \left(\frac{a^2 - c^2}{b^2}\right)^2 - 1 + 2\left(\frac{a^2 - c^2}{b^2}\right)^2\cos^2 \\ & + 2\left(\frac{b^2 - c^2}{b^2}\right)\cos^2\alpha \\ & - 4\left(\frac{a^2 + c^2}{b^2}\right)\cos\alpha\cos\beta\cos\gamma \\ & + 2\left(\frac{b^2 - a^2}{b^2}\right)\cos^2\gamma\Bigg] \end{aligned}$$

$$\begin{aligned} A_1 = 4\Bigg[& -\left(\frac{a^2 - c^2}{b^2}\right)\left(1 + \frac{a^2 - c^2}{b^2}\right)\cos\beta \\ & + \frac{2a^2}{b^2}\cos^2\gamma\cos\beta \\ & - \left(1 - \left(\frac{a^2 + c^2}{b^2}\right)\right)\cos\alpha\cos\gamma\Bigg] \end{aligned}$$

$$A_0 = \left(1 + \frac{a^2 - c^2}{b^2}\right)^2 - \frac{4a^2}{b^2}\cos^2\gamma.$$

which will give 4 slns for V. Take those two real solutions and substitute into eqn : to solve for u.

$$u = \frac{\left(-1+\frac{a^2-c^2}{b^2}\right)v^2-2\left(\frac{a^2-c^2}{b^2}\right)\cos\beta v+1+\frac{a^2-c^2}{b^2}}{2(\cos\gamma-v\cos\alpha)} \quad (8)$$

Then, choose (u, v) which are both positive.

Next, use:

$$s_1^2 = \frac{c^2}{1+u^2-2u\cos\gamma}$$

to get s1.

$$S_2 = u S_1$$

$$S_3 = V S_1 .$$

Step 5:

Find a, b, c in camera coordinate :

$$^{c}P_1 = S_1 \hat{j_1}$$

$$^{c}P_2 = S_2 \hat{j_2}$$

$$^{c}P_3 = S_3 \hat{j_3} .$$