Code 1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      PROBLEM 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


syms l1 l2 q1(t) q2(t) r M g F1 F2  q1_d(t) q1_dd(t) q2_d(t) q2_dd(t)

%%%%%%%%%%%%%%%%%%%%%%
% Symbol definitions
%%%%%%%%%%%%%%%%%%%%%%
% q(t) - Represents theta(t)
% F - Represents generalized force or torque
% q_d(t) - Represents derivative of q wrt to t
% q_dd(t)- Represents second derivative of q wrt to t
% r - Position of COM wrt to inertial(world) frame
% l - link length of robot

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Lagrangian
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r = [(l2*cos(q2)+l1)*cos(q1)
     (l2*cos(q2)+l1)*sin(q1)
     l2*sin(q2)]

v = diff(r);  %derivate of r
T = simplify(sym(1)/2* M * transpose(v)*v)
V = M*g*l2*sin(q2)
L = T-V



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    Taking derivatives
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Derivative of L wrt to q
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dL_dq1 = functionalDerivative(L,q1);
dL_dq2 = functionalDerivative(L,q2);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Derivative of L wrt q_dot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Replacing derivatives with symbols
% Matlab can't take functional derivative with
% respect to a derivative
L=subs(L,diff(q2(t),t),q2_d);
L=subs(L,diff(q1(t),t),q1_d);
```

```
dL_dq1_dot= functionalDerivative(L,q1_d);
dL_dq2_dot= functionalDerivative(L,q2_d);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Derivative of dl/dq_dot wrt to t
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Taking derivative of dL_dq with respect to t
dt_dL_dq1_dot = diff(dL_dq1_dot,t);
dt_dL_dq2_dot = diff(dL_dq2_dot,t);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Equation of Motion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Replacing derivatives with symbols
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Replacing derivatives with symbols to perform substaction
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q1_d(t),t),q1_dd);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q1(t),t),q1_d);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q2_d(t),t),q2_dd);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q2(t),t),q2_d);


dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q1_d(t),t),q1_dd);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q1(t),t),q1_d);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q2_d(t),t),q2_dd);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q2(t),t),q2_d);



% Replacing derivatives with symbols to perform substaction
dL_dq1=subs(dL_dq1,diff(q1(t),t,t),q1_dd);
dL_dq1=subs(dL_dq1,diff(q1(t),t),q1_d);
dL_dq1=subs(dL_dq1,diff(q2(t),t,t),q2_dd);
dL_dq1=subs(dL_dq1,diff(q2(t),t),q2_d);


dL_dq2=subs(dL_dq2,diff(q1(t),t,t),q1_dd);
dL_dq2=subs(dL_dq2,diff(q1(t),t),q1_d);
dL_dq2=subs(dL_dq2,diff(q2(t),t,t),q2_dd);
dL_dq2=subs(dL_dq2,diff(q2(t),t),q2_d);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%        Finding Equations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
eqn1 = simplify(dt_dL_dq1_dot - dL_dq1);
enq1=simplify(subs(eqn1,l1,2*l2))== F1

eqn2 =simplify(dt_dL_dq2_dot - dL_dq2);
eqn2 = simplify(subs(eqn2,l1,2*l2))== F2
```

Output

```
>> dynamicsProblem1

r(t) =

 cos(q1(t))*(l1 + l2*cos(q2(t)))
 sin(q1(t))*(l1 + l2*cos(q2(t)))
                  l2*sin(q2(t))


T(t) =

(M*(sin(q1(t))*(l1 + l2*cos(q2(t)))*diff(q1(t), t) + l2*cos(q1(t))*sin(q2(t))*diff(q2(t), t))^2)/2 + (M*(cos(q1(t))*(l1 + l2*cos(q2(t)))*diff(q1(t), t) - l2*sin(q1(t))*sin(q2(t))*diff(q2(t), t))^2)/2 +
(M*l2^2*cos(q2(t))^2*diff(q2(t), t)^2)/2


V(t) =

M*g*l2*sin(q2(t))


L(t) =

(M*(sin(q1(t))*(l1 + l2*cos(q2(t)))*diff(q1(t), t) + l2*cos(q1(t))*sin(q2(t))*diff(q2(t), t))^2)/2 + (M*(cos(q1(t))*(l1 + l2*cos(q2(t)))*diff(q1(t), t) - l2*sin(q1(t))*sin(q2(t))*diff(q2(t), t))^2)/2 +
(M*l2^2*cos(q2(t))^2*diff(q2(t), t)^2)/2 - M*g*l2*sin(q2(t))


enq1(t) =

2*M*l2*(cos(q2(t)) + 2)*(2*l2*q1_dd(t) + l2*cos(q2(t))*q1_dd(t) - 2*l2*sin(q2(t))*q1_d(t)*q2_d(t)) == F1


eqn2(t) =

2*M*l2^2*q2_dd(t) + 2*M*l2^2*sin(q2(t))*q1_d(t)^2 + M*g*l2*cos(q2(t)) + M*l2^2*cos(q2(t))*sin(q2(t))*q1_d(t)^2 == F2
```

Code 2

```python
import numpy as np

def transZ(n):
    a = np.identity(4)
    a[2,3]=n
    return a

def transX(n):
    a=np.identity(4)
    a[0,3]=n
    return a

def transY(n):
    a=np.identity(4)
    a[1,3]=n
    return a

def rotX(n):
    x=np.deg2rad(n)
    a=np.array([[1,0,0,0],
                [0,np.cos(x),-np.sin(x),0],
                [0,np.sin(x),np.cos(x),0 ],
                [0,  0,0,1]
```

```python
                ])
    return a

def rotY(n):
    x=np.deg2rad(n)
    a=np.array([[np.cos(x),0,np.sin(x),0],
                [0,1,0,0],
                [-np.sin(x),0,np.cos(x),0 ],
                [0, 0,0,1]
                ])
    return a

def rotZ(n):
    x=np.deg2rad(n)
    a=np.array([[np.cos(x),-np.sin(x),0,0],
                [np.sin(x),np.cos(x),0,0],
                [0,0,1,0 ],
                [0,0,0,1]
                ])
    return a


def A_dh(theta,d,a,alpha):
   return  np.dot(rotZ(theta),
           np.dot(transZ(d),
           np.dot(transX(a),
           rotX(alpha))))




print('##############################################################')
print('                      Question 2a                              ')
print('##############################################################')
print(' ')

# q = np.array([90, -30, 60, 2])
q= np.array([150, 45 , 30 , 2])
l1 = 1.5
l2 = 1
l3 = 0.5
A1=A_dh(q[0]-90,l1,0,-30)
A2=A_dh(q[1]-90,l2,0,90)
A3=A_dh(q[2]+60,0,0,-90)
AN=A_dh(0,q[3]+l3,0,0)

T0_1 = A1
T0_2 = np.dot(T0_1,A2)
T0_3 = np.dot(T0_2,A3)
T0_n = np.dot(T0_3,AN)


z0 = np.array([0, 0, 1, 0])
z1 = np.dot(T0_1,z0)[:3]
z2 = np.dot(T0_2,z0)[:3]
z3 = np.dot(T0_3,z0)[:3]
z0 = z0[:3]

O0 = np.array([0, 0, 0, 1])
```

```
O1 = np.dot(T0_1,O0)[:3]
O2 = np.dot(T0_2,O0)[:3]
On = np.dot(T0_n,O0)[:3]
O0 = O0[:3]



j1 = np.concatenate((np.cross(z0,(On - O0)),z0))
j2 = np.concatenate((np.cross(z1,(On - O1)),z1))
j3 = np.concatenate((np.cross(z2,(On - O2)),z2))
j4 = np.concatenate((z3,O0))




J = np.column_stack((j1,j2,j3,j4))
F = np.array([1, 0, 0, 0, 0, 0])
T = np.round(np.dot(J.T,F),3)

print('Joint torques ')
print(T)


print('##############################################################')
print('                       Question 2b                               ')
print('##############################################################')
print(' ')

T= np.array([2, 3, -1, 0.5])
J= J[0:3,:]
pseudoInv = np.dot(np.linalg.inv(np.dot(J,J.T)),J)
print(np.dot(pseudoInv,T))
```

Output

```
def rotX(n):
    x=np.deg2rad(n)
exam.py
ak@ubuntu16:~$ python3 exam.py
##############################################################
                      Question 2a
##############################################################

Joint torques
[ 0.515  0.442  1.083 -0.884]
##############################################################
                      Question 2b
##############################################################

[-0.33580664 -0.96356068  0.63309673]
ak@ubuntu16:~$
```

Code 3

APPROACH 1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       PROBLEM 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Note This approach doesn't account for Rotational Inertia'

syms I  m1 m2 m3 g F1 F2 F3
syms q1(t) q2(t) q3(t) q1_d(t) q1_dd(t) q2_d(t) q2_dd(t) q3_d(t) q3_dd(t)
syms a1c a2c a3c a1 a2 a3

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Symbol definitions
%%%%%%%%%%%%%%%%%%%%%%%%%%
% q(t) - Represents generalized coordinates
% F - Represents generalized force or torque
% q_d(t) - Represents derivative of q wrt to t
% q_dd(t)- Represents second derivative of q wrt to t
% r - Position of COM wrt to inertial(world) frame
% a's - robot link parameters
% I - Rotational Inertia from links

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Lagrangian
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
r1 = [a1c*cos(q1)
      a1c*sin(q1)];

r2 = [(a1+q2+a2c)*cos(q1)
      (a1+q2+a2c)*sin(q1)];

r3 = [(a1+ q2 + a2)*cos(q1) + a3c*cos(q3)
      (a1+ q2 + a2)*sin(q1) + a3c*sin(q3)];

% Getting velocities
v1 = diff(r1);
v2 = diff(r2);
v3 = diff(r3);

% Kinetic Energy
T1 = sym(1)/2 * transpose(v1) * v1;
T2 = sym(2)/2 * transpose(v2) * v2;
T3 = sym(3)/3 * transpose(v3) * v3;
```

```
T = T1 + T2 + T3

% Potential Energy
V1 = m1*g*a1c*sin(q1);
V2 = m2*g*(a1c + q2* + a2c)*sin(q1);
V3 = m3*g*((a1+q2*+a2)*sin(q1) + a3c*sin(q3));

V = V1 + V2 + V3

% Lagrangian
L = T-V


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Taking derivatives
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     Derivative of L wrt to q
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dL_dq1 = functionalDerivative(L,q1);
dL_dq2 = functionalDerivative(L,q2);
dL_dq3 = functionalDerivative(L,q3);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Derivative of L wrt q_dot
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Replacing derivatives with symbols
% Matlab can't take functional derivative with
% respect to a derivative
L=subs(L,diff(q1(t),t),q1_d);
L=subs(L,diff(q2(t),t),q2_d);
L=subs(L,diff(q3(t),t),q3_d);

dL_dq1_dot= functionalDerivative(L,q1_d);
dL_dq2_dot= functionalDerivative(L,q2_d);
dL_dq3_dot= functionalDerivative(L,q3_d);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  Derivative of dl/dq_dot wrt to t
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Taking derivative of dL_dq with respect to t
dt_dL_dq1_dot = diff(dL_dq1_dot,t);
dt_dL_dq2_dot = diff(dL_dq2_dot,t);
dt_dL_dq3_dot = diff(dL_dq3_dot,t);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      Equation of Motion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Replacing derivatives with symbols
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Replacing derivatives with symbols to perform substaction
```

```
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q1_d(t),t),q1_dd);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q1(t),t),q1_d);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q2_d(t),t),q2_dd);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q2(t),t),q2_d);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q3_d(t),t),q3_dd);
dt_dL_dq1_dot=subs(dt_dL_dq1_dot,diff(q3(t),t),q3_d);

dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q1_d(t),t),q1_dd);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q1(t),t),q1_d);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q2_d(t),t),q2_dd);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q2(t),t),q2_d);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q3_d(t),t),q3_dd);
dt_dL_dq2_dot=subs(dt_dL_dq2_dot,diff(q3(t),t),q3_d);

dt_dL_dq3_dot=subs(dt_dL_dq3_dot,diff(q1_d(t),t),q1_dd);
dt_dL_dq3_dot=subs(dt_dL_dq3_dot,diff(q1(t),t),q1_d);
dt_dL_dq3_dot=subs(dt_dL_dq3_dot,diff(q2_d(t),t),q2_dd);
dt_dL_dq3_dot=subs(dt_dL_dq3_dot,diff(q2(t),t),q2_d);
dt_dL_dq3_dot=subs(dt_dL_dq3_dot,diff(q3_d(t),t),q3_dd);
dt_dL_dq3_dot=subs(dt_dL_dq3_dot,diff(q3(t),t),q3_d);


% Replacing derivatives with symbols to perform substaction
dL_dq1=subs(dL_dq1,diff(q1(t),t,t),q1_dd);
dL_dq1=subs(dL_dq1,diff(q1(t),t),q1_d);
dL_dq1=subs(dL_dq1,diff(q2(t),t,t),q2_dd);
dL_dq1=subs(dL_dq1,diff(q2(t),t),q2_d);
dL_dq1=subs(dL_dq1,diff(q3(t),t,t),q3_dd);
dL_dq1=subs(dL_dq1,diff(q3(t),t),q3_d);

dL_dq2=subs(dL_dq2,diff(q1(t),t,t),q1_dd);
dL_dq2=subs(dL_dq2,diff(q1(t),t),q1_d);
dL_dq2=subs(dL_dq2,diff(q2(t),t,t),q2_dd);
dL_dq2=subs(dL_dq2,diff(q2(t),t),q2_d);
dL_dq2=subs(dL_dq2,diff(q3(t),t,t),q3_dd);
dL_dq2=subs(dL_dq2,diff(q3(t),t),q3_d);

dL_dq3=subs(dL_dq3,diff(q1(t),t,t),q1_dd);
dL_dq3=subs(dL_dq3,diff(q1(t),t),q1_d);
dL_dq3=subs(dL_dq3,diff(q2(t),t,t),q2_dd);
dL_dq3=subs(dL_dq3,diff(q2(t),t),q2_d);
dL_dq3=subs(dL_dq3,diff(q3(t),t,t),q3_dd);
dL_dq3=subs(dL_dq3,diff(q3(t),t),q3_d);



% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% %        Finding Equations
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
eqn1 = simplify(dt_dL_dq1_dot - dL_dq1) == F1
eqn2 = simplify(dt_dL_dq2_dot - dL_dq2) == F2
eqn3 = simplify(dt_dL_dq3_dot - dL_dq3) == F3
```

Output

```
>> dynamicsProblem3

T(t) =

(sin(q1(t))*diff(q2(t), t) + cos(q1(t))*(a1 + a2 + q2(t))*diff(q1(t), t) + a3c*cos(q3(t))*diff(q3(t), t))^2 + (sin(q1(t))*diff(q1(t), t)*(a1 + a2 + q2(t)) - cos(q1(t))*diff(q2(t), t) + a3c*sin(q3(t))*diff(q3(t), t))^2 +
(sin(q1(t))*diff(q2(t), t) + cos(q1(t))*(a1 + a2c + q2(t))*diff(q1(t), t))^2 + (cos(q1(t))*diff(q2(t), t) - sin(q1(t))*(a1 + a2c + q2(t))*diff(q1(t), t))^2 + (a1c^2*cos(q1(t))^2*diff(q1(t), t)^2)/2 +
(a1c^2*sin(q1(t))^2*diff(q1(t), t)^2)/2


V(t) =

g*m3*(a3c*sin(q3(t)) + sin(q1(t))*(a1 + a2*q2(t))) + a1c*g*m1*sin(q1(t)) + g*m2*sin(q1(t))*(a1c + a2c*q2(t))


L(t) =

(sin(q1(t))*diff(q2(t), t) + cos(q1(t))*(a1 + a2 + q2(t))*diff(q1(t), t) + a3c*cos(q3(t))*diff(q3(t), t))^2 + (sin(q1(t))*diff(q1(t), t)*(a1 + a2 + q2(t)) - cos(q1(t))*diff(q2(t), t) + a3c*sin(q3(t))*diff(q3(t), t))^2 +
(sin(q1(t))*diff(q2(t), t) + cos(q1(t))*(a1 + a2c + q2(t))*diff(q1(t), t))^2 + (cos(q1(t))*diff(q2(t), t) - sin(q1(t))*(a1 + a2c + q2(t))*diff(q1(t), t))^2 - g*m3*(a3c*sin(q3(t)) + sin(q1(t))*(a1 + a2*q2(t))) +
(a1c^2*cos(q1(t))^2*diff(q1(t), t)^2)/2 + (a1c^2*sin(q1(t))^2*diff(q1(t), t)^2)/2 - a1c*g*m1*sin(q1(t)) - g*m2*sin(q1(t))*(a1c + a2c*q2(t))


eqn1(t) =

8*a1^2*q1_dd(t) + 4*a2^2*q1_dd(t) + 2*a1c^2*q1_dd(t) + 4*a2c^2*q1_dd(t) + 8*q2(t)^2*q1_dd(t) + 8*a1*a2*q1_dd(t) + 8*a1*a2c*q1_dd(t) + 16*q2(t)*q1_d(t)*q2_d(t) + 16*a1*q2(t)*q1_dd(t) + 8*a2*q2(t)*q1_dd(t) + 16*a1*q1_d(t)*q2_d(t
+ 8*a2*q1_d(t)*q2_d(t) + 8*a2c*q2(t)*q1_dd(t) + 8*a2c*q1_d(t)*q2_d(t) + 4*a1*a3c*q3_d(t)^2*sin(q1(t) - q3(t)) + 4*a2*a3c*q3_d(t)^2*sin(q1(t) - q3(t)) + 4*a3c*q2(t)*q3_dd(t)*cos(q1(t) - q3(t)) + 2*a3c*q2_d(t)*q3_d(t)*cos(q1(t) -
q3(t)) + a1*g*m3*cos(q1(t)) + a1c*g*m1*cos(q1(t)) + a1c*g*m2*cos(q1(t)) + 4*a3c*q2(t)*q3_d(t)^2*sin(q1(t) - q3(t)) + 4*a1*a3c*q3_dd(t)*cos(q1(t) - q3(t)) + 4*a2*a3c*q3_dd(t)*cos(q1(t) - q3(t)) -
2*a3c*q2(t)*q1_d(t)*q3_d(t)*sin(q1(t) - q3(t)) + a2*g*m3*cos(q1(t))*q2(t) + a2c*g*m2*cos(q1(t))*q2(t) - 2*a1*a3c*q1_d(t)*q3_d(t)*sin(q1(t) - q3(t)) - 2*a2*a3c*q1_d(t)*q3_d(t)*sin(q1(t) - q3(t)) == F1


eqn2(t) =

8*q2_dd(t) - 4*a1*q1_d(t)^2 - 2*a2*q1_d(t)^2 - 2*a2c*q1_d(t)^2 - 4*q2(t)*q1_d(t)^2 + 4*a3c*q3_dd(t)*sin(q1(t) - q3(t)) - 4*a3c*q3_d(t)^2*cos(q1(t) - q3(t)) + 2*a3c*q1_d(t)*q3_d(t)*cos(q1(t) - q3(t)) + a2*g*m3*sin(q1(t)) +
a2c*g*m2*sin(q1(t)) == F2


eqn3(t) =

4*a3c^2*q3_dd(t) + 4*a3c*q2_dd(t)*sin(q1(t) - q3(t)) - 4*a1*a3c*q1_d(t)^2*sin(q1(t) - q3(t)) - 4*a2*a3c*q1_d(t)^2*sin(q1(t) - q3(t)) + 4*a3c*q2(t)*q1_dd(t)*cos(q1(t) - q3(t)) + 8*a3c*q1_d(t)*q2_d(t)*cos(q1(t) - q3(t)) -
2*a3c*q2_d(t)*q3_d(t)*cos(q1(t) - q3(t)) + a3c*g*m3*cos(q3(t)) - 4*a3c*q2(t)*q1_d(t)^2*sin(q1(t) - q3(t)) + 4*a1*a3c*q1_dd(t)*cos(q1(t) - q3(t)) + 4*a2*a3c*q1_dd(t)*cos(q1(t) - q3(t)) + 2*a3c*q2(t)*q1_d(t)*q3_d(t)*sin(q1(t) -
q3(t)) + 2*a1*a3c*q1_d(t)*q3_d(t)*sin(q1(t) - q3(t)) + 2*a2*a3c*q1_d(t)*q3_d(t)*sin(q1(t) - q3(t)) == F3
```

Approach 2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            PROBLEM 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% This approach uses christoffel symbols and accounts for rotational
% inertia


syms I1_xx I1_xy I1_xz I1_yx I1_yy I1_yz I1_zx I1_zy I1_zz
syms I2_xx I2_xy I2_xz I2_yx I2_yy I2_yz I2_zx I2_zy I2_zz
syms I3_xx I3_xy I3_xz I3_yx I3_yy I3_yz I3_zx I3_zy I3_zz

syms m1 m2 m3 g F1 F2 F3
syms q1(t) q2(t) q3(t) q1_d(t) q1_dd(t) q2_d(t) q2_dd(t) q3_d(t) q3_dd(t)
syms a1c a2c a3c a1 a2 a3


%%%%%%%%%%%%%%%%%%%%%%
% Symbol definitions
%%%%%%%%%%%%%%%%%%%%%%
% q(t) - Represents generalized coordinates
% F - Represents generalized force or torque
% m - Represents mass of links
% q_d(t) - Represents derivative of q wrt to t
% q_dd(t)- Represents second derivative of q wrt to t
% r - Position of COM wrt to inertial(world) frame
% a's - robot link parameters
% I - Rotational Inertia from links with respect to inertial frame
```

```
I1 = [I1_xx I1_xy I1_xz;
      I1_yx I1_yy I1_yz;
      I1_zx I1_zy I1_zz];

I2 = [I2_xx I2_xy I2_xz;
      I2_yx I2_yy I2_yz;
      I2_zx I2_zy I2_zz];

I3 = [I3_xx I3_xy I3_xz;
      I3_yx I3_yy I3_yz;
      I3_zx I3_zy I3_zz];

Jvc1 =  [-a1c*sin(q1) 0 0;
          a1c*cos(q1) 0 0;
          0           0 0];

Jvc2 = [-sin(q1)*(a1+q2+a2c) cos(q1) 0;
         cos(q1)*(a1+q2+a2c) sin(q1) 0;
         0                   0      0];

Jvc3 = [-sin(q1)*(a1+q2+a2) cos(q1) -a3c*sin(q3);
         cos(q1)*(a1+q2+a2) sin(q1)  a3c*cos(q3);
         0                  0        0];

Jw1 =  [0 0 0;
        0 0 0;
        1 0 0];

Jw2 =  [0 0 0;
        0 0 0;
        0 0 0];

Jw3 = [ 0 0 0;
        0 0 0;
        1 0 1];

D =  m1*transpose(Jvc1)*Jvc1 + m2*transpose(Jvc2)*Jvc2 ...
   + m3*transpose(Jvc3)*Jvc3 + transpose(Jw1)*I1*Jw1   ...
   + transpose(Jw2)*I2*Jw2 + transpose(Jw3)*I3*Jw3 ;
D = simplify(D); % inertia matrix

P =     m1*g*a1c*sin(q1) + m2*g*(a1+q2+a2c)*sin(q1) ...
      + m3*g*((a1+q2+a2)*sin(q1) + a3c*sin(q3));

q = [q1; q2; q3];


Dmat = D(t); %This is necessary to index values of D
qmat = q(t); %This is necessary to index values of q

syms C [3 3 3]; %Christofel Symbols is a 3x3x3 matrix

%calculation christofell symbols
for i = 1:3
    for j = 1:3
        for k= 1:3
              C(i,j,k) = functionalDerivative(Dmat(k,j),qmat(j)) ...
                       + functionalDerivative(Dmat(k,i),qmat(i)) ...
```

```
                            - functionalDerivative(Dmat(i,j),qmat(k));
                C(i,j,k) = sym(1)/2 * C(i,j,k);
            end
        end
    end

    C = simplify(C);

    syms dP [3 1]; %derivate of potential energy wrt q's

    for i = 1:3
        dP(i)= functionalDerivative(P,qmat(i));
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %          Printing values
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    D
    C
    dP
```

Output

```
>> dynamicsProblem3Approach2

D(t) =

[ I1_zz + I3_zz + m2*q2(t)^2 + m3*q2(t)^2 + a1^2*m2 + a1^2*m3 + a2^2*m3 + a1c^2*m1 + a2c^2*m2 + 2*a1*m2*q2(t) + 2*a1*m3*q2(t) + 2*a2*m3*q2(t) + 2*a2c*m2*q2(t) + 2*a1*a2*m3 + 2*a1*a2c*m2,                        0, I3_zz + a3c*m3*cos(q1(t) - q3(t))*(a1 + a2 + q2(t))]
[                                                                                                                                                                                              0,                  m2 + m3,          a3c*m3*sin(q1(t) - q3(t))]
[                                                                                                  I3_zz + a3c*m3*cos(q1(t) - q3(t))*(a1 + a2 + q2(t)), a3c*m3*sin(q1(t) - q3(t)),                    m3*a3c^2 + I3_zz]


C(:,:,1) =

[                                           0,                                                                              0,                             a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t))]
[                                           0,                                                                              0, (a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)))/2 - (a3c*m3*cos(q1(t) - q3(t)))/2]
[ a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)), (a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)))/2 - (a3c*m3*cos(q1(t) - q3(t)))/2,                             a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t))]


C(:,:,2) =

[ - a1*m2 - a1*m3 - a2*m3 - a2c*m2 - m2*q2(t) - m3*q2(t),                              0,     -a3c*m3*cos(q1(t) - q3(t))]
[                                                      0,                              0, -(a3c*m3*cos(q1(t) - q3(t)))/2]
[                             -a3c*m3*cos(q1(t) - q3(t)), -(a3c*m3*cos(q1(t) - q3(t)))/2,     -a3c*m3*cos(q1(t) - q3(t))]


C(:,:,3) =

[     -a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)), -(a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)))/2, -a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t))]
[ -(a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)))/2,                                                 0,               (a3c*m3*cos(q1(t) - q3(t)))/2]
[     -a3c*m3*sin(q1(t) - q3(t))*(a1 + a2 + q2(t)),                     (a3c*m3*cos(q1(t) - q3(t)))/2,                                            0]


dP =

 g*m2*cos(q1(t))*q2(t) + g*m3*cos(q1(t))*q2(t) + a1*g*m2*cos(q1(t)) + a1*g*m3*cos(q1(t)) + a2*g*m3*cos(q1(t)) + a1c*g*m1*cos(q1(t)) + a2c*g*m2*cos(q1(t))
                                                                                                                   g*m2*sin(q1(t)) + g*m3*sin(q1(t))
                                                                                                                                a3c*g*m3*cos(q3(t))

>>
```