Introduction

My function used python to simulate the trajectory of the robot. In my function I itterativly call Forward dynamics and Euler step from the MR library as well as F_from_spring, which is used to calculate the force due to the spring and is included in the code.

Part 1

For the good timestep I picked the inputs:

```
puppet(thetalist=[0, 0, 0, 0, 0, 0], dthetalist=[0, 0, 0, 0, 0, 0], g=(0, 0, -9.8), Mlist=Mlist,
Slist=Slist, Glist=Glist, t=5, dt=.0025, damping=0, stiffness=0, springPos=[0, 0, 0], restLength=0)
```

and the bad times step was :

```
puppet(thetalist=[0, 0, 0, 0, 0, 0], dthetalist=[0, 0, 0, 0, 0, 0], g=(0, 0, -9.8), Mlist=Mlist,
Slist=Slist, Glist=Glist, t=5, dt=.01, damping=0, stiffness=0, springPos=[0, 0, 0], restLength=0)
```

Part 2

Yes, when damping is a large positive value, the simulation will not be able to run. This is because we input damping as a joint torque and when it is very large it actually begins to accelerate the robot in the opposite direction for a moment instead of slowing it down. Because we are using Euler step this begins to go crazy and the energy becomes too much to simulate. This could be mitigated by a smaller time step because then the torques at the joint would be calculated more often so they would more accurately represent a resistive friction. My inputs for positive tourge which slowed down the Robot are:

```
puppet(thetalist=[0, 0, 0, 0, 0, 0], dthetalist=[0, 0, 0, 0, 0, 0], g=(0, 0, -9.8), Mlist=Mlist,
Slist=Slist, Glist=Glist, t=5, dt=.01, damping=1, stiffness=0, springPos=[0, 0, 0], restLength=0)
```

and negative is:

```
puppet(thetalist=[0, 0, 0, 0, 0, 0], dthetalist=[0, 0, 0, 0, 0, 0], g=(0, 0, -9.8), Mlist=Mlist,
Slist=Slist, Glist=Glist, t=5, dt=.01, damping=-.01, stiffness=0, springPos=[0, 0, 0], restLength=0)
```

Part 3

For this simulation the data mostly made sense to me. Energy should have been conserved but as we saw in the second part of part 1, a timestep of .01 actually leads to an energy gain when using Euler step, because of this the energy is not actually conserved, there is a net gain. At very large stiffness this is even more evident because the instantaneous Ftip is so high that is causes accelerations in a certain direction for too long which makes the robot oscillate then gain too much energy and go crazy.

My input for part 3a was:

```
puppet(thetalist=[0, -1, 0, 0, 0, 0], dthetalist=[0, 0, 0, 0, 0, 0], g=(0, 0, 0), Mlist=Mlist, Slist=Slist,
Glist=Glist, t=10, dt=.01, damping=0, stiffness=25, springPos=[0, 0, 1], restLength=0)
```

for part 3b my input was:

```
puppet(thetalist=[0, -1, 0, 0, 0, 0], dthetalist=[0, 0, 0, 0, 0, 0], g=(0, 0, 0), Mlist=Mlist, Slist=Slist,
Glist=Glist, t=10, dt=.01, damping=2, stiffness=15, springPos=[0, 0, 1], restLength=0)
```