1.) a.) radius of circular motion $= r = \frac{v^2}{a}$

$F = ma \therefore a = \frac{F}{m} \therefore \frac{1}{a} = \frac{m}{F}$

$r = \frac{v^2 m}{F}$ where $F = qvB \therefore r = \frac{v^2 m}{qvB}$

$r = \frac{vm}{qB}$

$r = \frac{10^6 \times 9.1 \times 10^{-31}}{-1.6 \times 10^{-19} \times -2}$

$r = 2.84 \times 10^{-6} m$

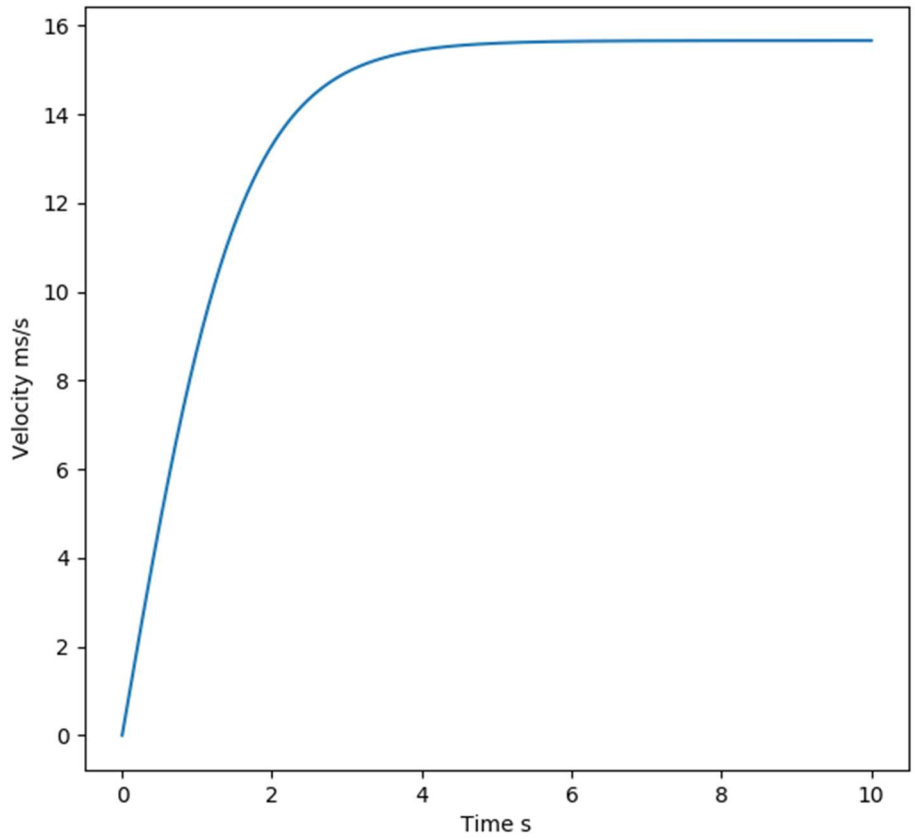
b.) $W = F \cdot d$ = where F is a vector, and d is the displacment vector. This means W is the dot product of F and d, and since F and d are perpendicular meaning their dotproduct is always zero and word done is zero.

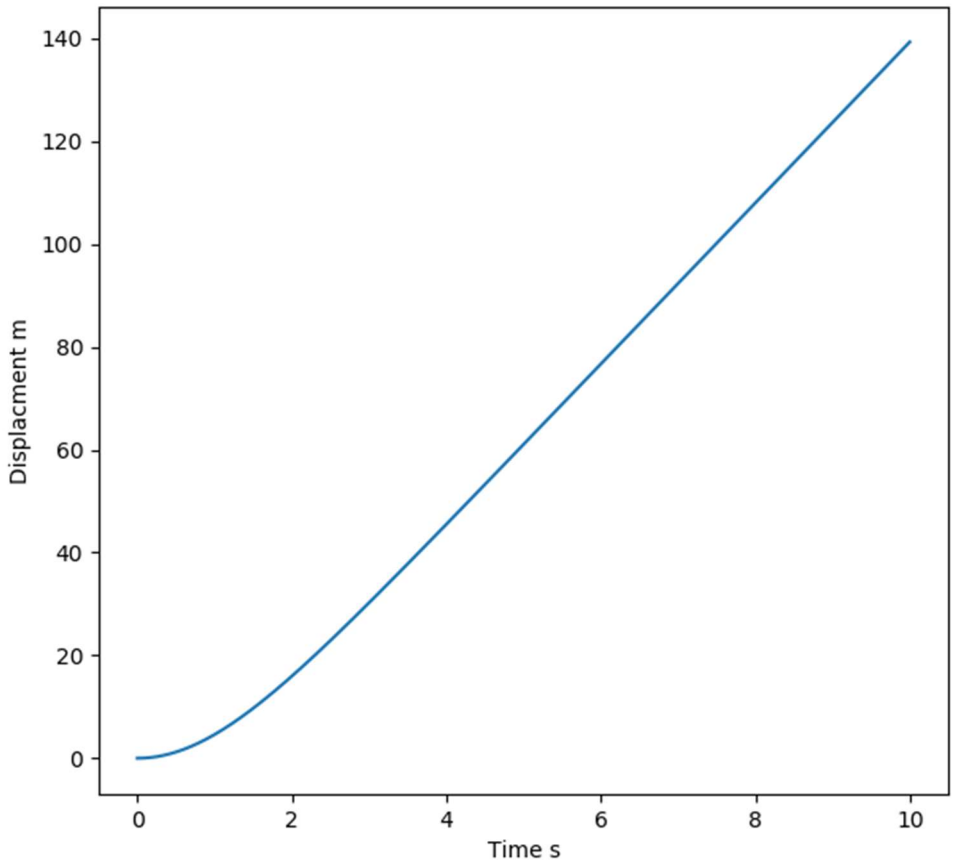
```
import math
import matplotlib.pyplot as plt

count = 0
mass = 5*(10**-4)
dragConstant = 2*(10**(-5))
tFinal = 10.0
numPts = 10000
acceleration = 0.0
previousVelocity = 0.0
totaldisplacment = 0.0
forceGravity = mass * 9.81

# Clacualte the terminal velocity of the hailstone as ait falls.
terminalVelocity = math.sqrt(forceGravity/dragConstant)

times = [] #array of times
velocityList = [] #array of velocities
displacments = [] #array of displacments

for x in xrange(numPts):
    times.append(x * (tFinal/numPts)) # Create an list of times

for x in xrange(numPts):
    # change in velocity
    deltaVelocity = acceleration * (tFinal/numPts)
    # current velocity
    velocity = previousVelocity + deltaVelocity
    # drag netForce using given velocity
    drag = dragConstant * velocity**2
    # total netForce on object (Fg - Fd)
    netForce = forceGravity - drag
    # acceleration on object
    acceleration = netForce / mass
    # previously calculated velocity used to find next velocity
    previousVelocity = velocity
    # add velocity to a list
    velocityList.append(velocity)
    # once the velocity is greater than 10% of the terminal velocity, print out
the time at that point, only once.
    if velocity > (0.9*terminalVelocity) and count == 0:
        print ('The time to reach 10% of the terminal velocity is
{}\n'.format(times[x]))
        count = count+1
count = 0

# Displacment = velocity * time

for x in xrange(numPts):
    # The velocity at time x
    velocity = velocityList[x]
    # Time spent at that velocity
    time = (tFinal/numPts)
    # displacment over that time period
    displacment = velocity * time
    # Total displacment
    totaldisplacment += displacment
    # Add each displacment to an array
    displacments.append(totaldisplacment)
```

```
    # once the velocity is greater than 10% of the terminal velocity, print out
the displacment at that point, only once.
    if velocity > (0.9*terminalVelocity) and count == 0:
        print ('The displacment at 10% of the terminal velocity is
{}\n'.format(displacments[x]))
        count = count+1

print ('the velocity after {} seconds is {}' .format(tFinal, previousVelocity))
print ('the displacment after {} seconds is {}' .format(tFinal, totaldisplacment))

plt.subplot(1,2,1)
plt.plot(times,velocityList)
plt.ylabel('Velocity ms/s')
plt.xlabel('Time s')

plt.subplot(1,2,2)
plt.plot(times, displacments)
plt.ylabel('Displacment m')
plt.xlabel('Time s')
plt.show()
```

When running this code, 3 arrays are created, one containing all the times that velocity and displacement will be calculated with. One with all the velocities, and then one with all the displacements.

These three arrays will then be plotted against each other providing these graphs.





It also prints out the time and displacement at which 90% of the terminal velocity is reached.

```
The time to reach 10% of the terminal velocity is 2.35 seconds

The displacement at 10% of the terminal velocity is 20.77 m

the velocity after 10.0 seconds is 15.6 ms^-1
the displacement after 10.0 seconds is 139.27 m
```

```
C:\Python27\python.exe
The time to reach 10% of the terminal velocity is 2.35

The displacment at 10% of the terminal velocity is 20.7658633622

the velocity after 10.0 seconds is 15.6603469546
the displacment after 10.0 seconds is 139.272982273

Process returned 0 (0x0)        execution time : 2.572 s
Press any key to continue . . .
```