Question A.1:

I installed Ubuntu 20.04 and ros noetic in Oracle VB.

Question A.2:

Below are a couple of screenshots of the obtained results. The code can be found in the repo.

```
stiggy@stiggy-VB:~/catkin_ws/src/q1/src$ roslaunch q1 lnch.launch
... logging to /home/stiggy/.ros/log/3d18f73a-c0b0-11ec-82ee-f1be90c0b255/roslaunch-stiggy-VB-3357.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://stiggy-VB:45783/

SUMMARY
========

PARAMETERS
 * /rosdistro: noetic
 * /rosversion: 1.15.14

NODES
  /
    listener (q1/subscriber.py)
    talker (q1/publisher.py)

ROS_MASTER_URI=http://localhost:11311

process[talker-1]: started with pid [3371]
process[listener-2]: started with pid [3372]
Team Abhiyaan rocks
Team Abhiyaan rocks
Team Abhiyaan rocks
Team Abhiyaan rocks
Team Abhiyaan rocks
Team Abhiyaan rocks
Team Abhiyaan rocks
Team Abhiyaan rocks
```

```
stiggy@stiggy-VB:~/catkin_ws$ rosrun q1 subnpub.py
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
maeT naayihbA skcor
```

Question A.3:

I was unable to complete this question. My unfinished attempt can be found in the repo. However, I had two ideas on how to solve this:

1. In the first method, we discretise the 2 body problem into small timesteps. In each timestep the turtles will move in a straight line, their directions and speeds determined by their velocity vectors. At the end of each timestep, the velocity vectors will be updated according to the acceleration vector ($\Delta V = A \Delta t$). These updated velocity vectors will then be used for the next timestep.
2. In the second method, we compute the equation of the elliptic path followed by the turtles. This can be done rather trivially using Lagrangian mechanics and the reduced mass concept. We then sample (say) 100 points on each of the ellipses and move the turtles from point to point at speeds determined by the solution to the E-L equation. Albeit a cool method, the validity of this approach as a ‘simulation’ is questionable.

Question B.1:

The code can be found in the repo. Implemented using python.

Question B.2:

The .ipynb for this question is in the repo. I have added comments to hopefully explain my implementation. It is also quite inefficient and hence takes a while to analyze each frame of the video, which being in 60 fps, makes it even worse. Basic image processing techniques were used in my code.

Question B.3.1:

1. **Cameras**: The most common sensor in autonomous vehicles are cameras. Multi camera solutions are typically used (eg. in teslas) due to their various advantages. Multiple cameras are used together to give a 3D image and a depth perspective which is not possible iif a single camera is used. Cameras also provide color information which is critical in understanding road signs, traffic lights, vehicle lights etc. Cameras also have high resolution, are cheap, and image processing techniques based on camera images have been well researched. The biggest disadvantage is the fact that cameras are highly dependent on environmental factors. Stormy, foggy weather or even just a really dark alley can pretty much render them useless.
2. **Radar sensors**: These are based on radio waves and are a very reliable way of obtaining distance information. They are unaffected by weather phenomena and have a

large range due to the nature of radio waves (they hardly scatter). The primary disadvantage is lack of high resolution and color information.

3. **Lidar sensors**: The working principle is identical to that of radar except that high frequency EM waves, namely, visible or UV light is used. Lidar is extremely useful in situations where 3D mapping of the environment is necessary due to its high resolution capacity. It is also independent of environmental factors (if uv light is used). The main disadvantage is the cost of these sensors.
4. **Ultrasonic sensors**: These are also similar to radar/lidar except that they use sound waves instead of electromagnetic radiation. They excel in the short range and are very inexpensive. However due to their inability to scale over large distances, lack of resolution and color information, these sensors are mostly only used as parking sensors.

As we have seen, each sensor has its own advantages and disadvantages and none of them are the perfect solution for all situations and applications.This is why almost all autonomous solutions possess multiple of these sensors, which, in union, are able to compensate for the weakness of other sensors and form a single model of the environment (sensor fusion).

My source for this question was primarily this.

Question B.3.2:

The paper proposes a system to detect cross-roads, both T-cross and + -cross, and distinguish between them by acquiring 3D point clouds using a dense 64-beam scanning LIDAR mounted on the roof of the vehicle. Here, intersection recognition is treated as a classification problem. Situations ahead of a vehicle are classified as intersection and road segment, and in addition, distinction is made between *T*-shaped and +-shaped intersections.

Drawbacks in video-based systems: For autonomous driving, video-based methods were proposed earlier. However, poor lighting conditions such as overcasts, overexposure and other interference from moving vehicles and pedestrians make video-based methods an extremely difficult and near impossible task even if sophisticated image processing techniques are used at the expense of processing speed.

Key contribution of this work: (1) Treating intersection recognition as a classification problem. (2) The beam width is a little wider than the width of the vehicle; all rays which do not end in an obstacle are more likely to be in the drivable region. Based on this, the features extracted from the beam model are more effective. (3) The launching point of the beam is an adaptive distance in front of the vehicle instead of a fixed distance. This distance is directly proportional to vehicle speed. This would meet the demands of autonomous driving.

The algorithm: (1) Convert point cloud to grid map, (2) Use thresholding to create a binary image of the grid map, (3) detect pedestrians and other vehicles using image processing, and clear those cells from the grid, (4) use the resulting grid and run a machine learning (Support Vector Machine) algorithm for intersection recognition (classification problem).

Situations where this method might not work: This paper proposes a method that uses lidar to form an overhead view of the intersection. An obvious situation where it might fail (for example) is roads in a field where the distinction between the tarmac and grass is undetectable by lidar due to its inability to detect color. Also, any intersection where the normalized length mentioned in the paper is similar in all directions (like an intersection with no buildings nearby for light to bounce off), will be a problem for this system.