- ■ What was the goal of your project?

Learning a classical robotic localization algorithm called the particle filter by building a reliable and efficient particle filter program.

- ■ How did you solve the problem? (Note: this doesn't have to be super-detailed, you should try to explain what you did at a high-level so that others in the class could reasonably understand what you did).

    a. initialize particles based on a distribution centered around a "2D Pose Estimate" of the robot's position in the map
    b. update particles' pose and weights by robot's sensor data
    c. recalculate robot's pose from particles' pose
    d. resample particles based on their weights

- ■ Describe a design decision you had to make when working on your project and what you ultimately did (and why)? These design decisions could be particular choices for how you implemented some part of an algorithm or perhaps a decision regarding which of two external packages to use in your project.

Note: this question seems to be an artifact of last years project, which was more open ended and did not contain scaffolded code.
Nonetheless, we did have some design choice in how we initialized the particles. We chose to sample from a gaussian distribution for the x, y location, and a uniform distribution for the yaw. Since we were not hoping to solve the localization problem of the robot finding itself in an unknown map, we refrained from figuring out how to initialize particles evenly distributed about every grid square of the map.

- ■ What if any challenges did you face along the way?
    - ○ Robotic Issues
        - ■ Transferring x, y and theta between different coordination systems.
        - ■ In general, it took us a long time to understand the conceptual nuances of the robot geometry (such as the occupancy field in update_particles_with_laser)
    - ○ Pythonic issues
        - ■ The use of arctan vs arctan2 in getting correctly oriented yaw calculations
        - ■ Trying to generalize the number of angles from laser scan data we use (lesson: One cannot index arrays by floats!)

■ What would you do to improve your project if you had more time?

Trying to find the most efficient combination among the number of particles, number of scan angles and the computational efficiencies to improve the overall accuracy and efficiency of the program.

■ Did you learn any interesting lessons for future robotic programming projects? These could relate to working on robotics projects in teams, working on more open-ended (and longer term) problems, or any other relevant topic.

Working and testing the program in a simulated system can be very convenient and helpful sometimes. For future projects, getting a gazebo simulation working first is a must.
Setting up the code architecture, without filling in the implementations, was a very effective way to make sure all the systems were communicating correctly. This would be a good strategy for future robotic software projects with many pieces.
The code architecture diagram helped in conceptualizing how all the pieces moved together. Utilizing the whiteboards to clearly communicate with your teammate was a definite must.
Using double monitors was also helpful for our sanity! To see the simulation map reality at the same time with the rviz map.