Homework 5: Car Tracking

Please keep the title of each section and delete examples.

Part I. Implementation (15%):

- **Please screenshot your code snippets of Part 1 ~ Part 3, and explain**

- **Part 1**

```python
def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
    # BEGIN_YOUR_CODE
    for row in range(self.belief.numRows):
        for col in range(self.belief.numCols):
            # Calculate the distance between the agent and the tile
            dist = math.dist( ( util.colToX(col), util.rowToY(row) ), (agentX, agentY))
            # Update the belief
            probability = self.belief.getProb(row, col) * util.pdf(dist, Const.SONAR_STD, observedDist)
            self.belief.setProb(row, col, probability)
    # Normalize the belief
    self.belief.normalize()
    # END_YOUR_CODE
```

- **Part 2**

```python
def elapseTime(self) -> None:
    if self.skipElapse: ### ONLY FOR THE GRADER TO USE IN Part 1
        return
    # BEGIN_YOUR_CODE
    # Create a new belief
    newBeilf = util.Belief(self.belief.getNumRows(), self.belief.getNumCols(), value=0)
    for (oldTile, newTile), transProb in self.transProb.items():
        # Get the row and col of the new tile
        row = newTile[0]
        col = newTile[1]
        # Calculate the probability of the new tile
        probability = self.belief.getProb(oldTile[0], oldTile[1]) * transProb
        # Add the probability to the new belief
        newBeilf.addProb(row, col, probability)
    # Update and Normalize the belief
    self.belief = newBeilf
    self.belief.normalize()
    # END_YOUR_CODE
```

- **Part 3-1**

```python
def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
        # BEGIN_YOUR_CODE
        reweight = dict()
        for (row, col), num in self.particles.items():
            dist = math.dist( ( util.colToX(col), util.rowToY(row) ), (agentX, agentY))
            num_particles = self.belief.getProb(row, col)
            density = util.pdf(dist, Const.SONAR_STD, observedDist)
            reweight[(row,col)] =  num_particles * density

        resample = dict()
        for i in range(self.NUM_PARTICLES):
            # Create a new dictionary during re-sampling the particles
            particle = util.weightedRandomChoice(reweight)
            # Calculate the number of particles at that grid square
            if particle in resample:
                resample[particle] += 1
            else:
                resample[particle] = 1
        # Update self.particles
        self.particles = resample
        # END_YOUR_CODE

        self.updateBelief()
```

- **Part 3-2**

```python
def elapseTime(self) -> None:
    # BEGIN_YOUR_CODE
    # Assigned a default value of 0 to each key
    proposal = collections.defaultdict(int)
    # Loop over particles
    for particle, num in self.particles.items():
        # If there are multiple particles at a particular location
        for i in range(num):
            # Sample a new particle location for each of particles
            x = util.weightedRandomChoice(self.transProbDict[particle])
            if x in proposal:
                proposal[x] += 1
            else:
                proposal[x] = 1
    self.particles = proposal
    # END_YOUR_CODE
```

Part II. Question answering (5%):

- **Please describe problems you met and how you solved them**
- **One of the problems is that I do not sure which variable or function to use, even though I understand what type of data is required. After searching and reading throughout the files, I was able to find the correct utilities to use.**