Homework 5: Car Tracking

Please keep the title of each section and delete examples.

Part I. Implementation (20%):

Part 1

```
# the other car is not moving
for col in range(self.belief.getNumCols()): # for every column
    for row in range(self.belief.getNumRows()): # for every row
        current = self.belief.getProb(row, col) # take the current probability of the location of the other car
        x = util.colToX(col) # converting column indices into locations
        y = util.rowToY(row) # converting row indicies into locations
        distance = math.sqrt(math.pow((agentX - x), 2) + math.pow((agentY - y), 2)) # find the distance between two cars
        probDensity = util.pdf(distance, Const.SONAR_STD, observedDist) # compute with the probability density function
        self.belief.setProb(row, col, current*probDensity) # update the probability
self.belief.normalize() # normalize
```

Part 2

```
# the other car is moving
newBelieve = util.Belief(self.belief.getNumRows(), self.belief.getNumCols(), 0) # create a new believe object
current = {} # also create a list to record the current probabilities
for col in range(newBelieve.getNumCols()): # for every column
    for row in range(newBelieve.getNumRows()): # for every row
        current[(row, col)] = self.belief.getProb(row, col) # record each probability
for tiles in self.transProb.keys(): # for every key of the transProb dictionary
    (oldTile, newTile) = tiles # split the pair into the old tile and the new tile
    newBelieve.addProb(newTile[0], newTile[1], current[oldTile]*self.transProb[tiles]) # add the value to the newTile
newBelieve.normalize() # normalize
self.belief = newBelieve # update the version
```

Part 3-1

```
newProb = collections.defaultdict(float) # create a new dictionary to record the new probabilities
for location in self.particles.keys(): # for every particle
    (row, col) = location # split the locations into rows and columns
    current = self.particles[location] # record each particle
    x = util.colToX(col) # converting column indices into locations
    y = util.rowToY(row) # converting row indices into locations
    distance = math.sqrt(math.pow((agentX - x), 2) + math.pow((agentY - y), 2)) # calculate the distance between particles and the agent
    probDensity = util.pdf(distance, Const.SONAR_STD, observedDist) # compute with the probability density function
    newProb[location] = current * probDensity # update the probability
newParticles = collections.defaultdict(int) #  create a new dictionary to record the newly calculated particles
for run in range(self.NUM_PARTICLES): # runs the number of particle times
    Location = util.weightedRandomChoice(newProb) # select an element randomly
    newParticles[Location] += 1 # add 1 to the particle being chosen
self.particles = newParticles # update particles
```

Part 3-2

```
newParticles = collections.defaultdict(int) # create a new dictionary to record the newly calculated particles
for location in self.particles: # for every particle
    for i in range(self.particles[location]):
        Location = util.weightedRandomChoice(self.transProbDict[location]) # select an element randomly
        newParticles[Location] += 1 # add 1 to the particle being chosen
self.particles = newParticles # update particles
```

Part II. Problems Encountered

I had trouble with the usage of dictionary, I think the problem was I didn't read the notes clearly, but I succeeded after I realized that there were pairs in the keys of the dictionaries.