Part 1

```
for row in range(self.belief.numRows):
    for col in range(self.belief.numCols):
        mean = math.dist((util.colToX(col), util.rowToY(row)), (agentX, agentY))
        Et = util.pdf(mean, Const.SONAR_STD, observedDist)
        self.belief.setProb(row, col, Et * self.belief.getProb(row, col))
self.belief.normalize()
```

Goal : $\mathbb{P}(H_t|E_1 = e_1, \cdots, E_{t-1} = e_{t-1})p(e_t|h_t)$

1. Firstly calculate Et by pdf of Gaussian distribution using util.pdf(mean, std, observedDist), where mean = distance ((X_tile,Y_tile) , (agentX, agentY)).
2. Secondly multiply self.belief.getProb() = $\mathbb{P}(H_t|E_1 = e_1, \cdots, E_{t-1} = e_{t-1})$ with Et.
3. Finally do normalization on the probabilities.

Part 2

```
tmp_belief = util.Belief(self.belief.getNumRows(), self.belief.getNumCols(), value=0)
for key, trans_Prob in self.transProb.items():
    (oldTile, newTile) = key
    new_prob = self.belief.getProb(oldTile[0], oldTile[1]) * trans_Prob
    tmp_belief .addProb(newTile[0], newTile[1], new_prob)
self.belief = tmp_belief
self.belief.normalize()
```

Goal : $\sum_{h_t} \mathbb{P}(H_t = h_t|E_1 = e_1, \cdots, E_t = e_t)p(h_{t+1}|h_t)$

1. To avoid using one updated grid to update another one, initialize a new belief by tmp_belief = util.Belief().
2. Then, for (key, value) = (oldTile, newTile), transProb) in self.tansProb{ }, update the current probability with transition probability such that new_prob = self.belief.getProb(oldTile[0], oldTile[1]) * trans_Prob.
3. Finally, do normalization on the probabilities.

Part 3-1 observe()

```
reweighted_particles = {}
for key, num_particle in self.particles.items():
    (row, col) = key
    dist = math.dist((agentX, agentY),(util.colToX(col), util.rowToY(row)))
    Et = util.pdf(dist, Const.SONAR_STD, observedDist)
    reweighted_particles[key] = self.particles[key] * Et

resample_particles = {}
for i in range(self.NUM_PARTICLES):
    particle = util.weightedRandomChoice(reweighted_particles)
    if particle in resample_particles:
        resample_particles[particle] += 1
    else:
        resample_particles[particle] = 1
self.particles = resample_particles
```

1. In the Reweight Part, firstly calculate the emission probability Et as part 1 ; then create a dictionary reweighted_particle{ } to store the particles weighted by Et.
2. (1) In the Resample Part, iterate |self.NUM_PARTICLES| times to create resample particles chosen by weightedRandomChoice(reweighted_particle).
   (2) Next, store those particles in dictionary resample_particles{ } with conditional statement “particle in resample_particles : …” to avoid key errors.
   (3) Lastly self.particles = resample_particles

Part 3-2 elapseTime()

```
proposal = collections.defaultdict(int)
for particle in self.particles:
    num = self.particles[particle]
    for _ in range(num):
        chosen_particle = util.weightedRandomChoice(self.transProbDict[particle])
        if chosen_particle in proposal:
            proposal[chosen_particle] += 1
        else:
            proposal[chosen_particle] = 1

self.particles = proposal
```

1. For each particle (locations) in self.particles, there are “num” number of particles.
2. Secondly call chosen_particle = util.weightedRandomChoice() once for each particle in order to resample on the basis of transProbDict[particle].
3. Thirdly record the number of particles at each chosen_particle (location) in proposal{ }, and set self.particles = proposal

Problem

Q : In part 3-2, if I declare a dictionary like “proposal = { } “, Key error will be invoked even though conditional statement “if key in dictionary : …” is in the presence.

A : I still don’t know why the conditional statement works well on Part 3-1 but fail on Part 3-2, yet it can be solved by using defaultdict to declare a dictionary.