Q1:

In the test case 1, Ghost is random moving ghost ("SeededRandomGhostAgent"), Hence movement of this ghost in randomized. We are getting probabilities by usage of getPositionDistribution function. This function gives us probability about ghost next position distributions regardless of ghost real position. Therefore, the possibility of each legal position is same. In the second case Ghost does not use random, instead it is using "GoSouthAgent". Hence the ghost tends to move south. As time elapses possibilities of southern legal positions increases.

Q2:

In this question we update our beliefs after each observation. In every step we multiply our current belief with new observation probabilities. Hence, we can update our beliefs by getting different observations. In the test case 2, our agent cannot move. That's why observation of agent is same at all steps. Since pacman cannot observe corner points we cannot find ghost exact position. However, in the 3rd test case, our agent can move around positions and can get observations about corner points. After updating our belief, we can eliminate other positions for ghost. Then find it easily.

Q3:

Initially we are initializing particles uniformly. Hence in the we see lots of possible squares. As time passes, we update our particle observations. Particles are gathered with weights. At some point, the agent checks all the possible positions according to the particles. Then every particle turns into zero weight. That means agent cannot infer anything. When all particles receive zero weight, we reinitialize them uniformly. I tried to increase number of particles from 300 to until 5000, still I got re-initialization of particles. Increasing number of particles will not help us because time elapse is not used.

Q4:

Exact inference gives better results in terms of accuracy than compared to approximate inference. In exact inference we are using real data. Hence, we are calculating all possibilities. However, in approximate inference we use samples then gets probabilities on them. That's why approximate inference gives worse results. On the other hand, approximate inference has advantage on calculation time. Increasing number of particles make sense for approximate inference if time elapse is included. By this way we can get enough accuracy to find ghosts while keeping calculation cost minimum.

Q5:

In this Question implementation is similar to the approximate inference. However, this time we have different ghosts which are dependent each other. Hence while calculating new distributions, we are going to use all old ghost particles.

To initialize I created cartesian matrix itertools.product for each ghost. Then we shuffle them in order to ensure even placement of particles across the board.

In elapse time function, each particle is list of ghost beliefs since we use all ghosts for each particle.

```
for oldParticle in self.particles:
    newParticle = list(oldParticle)  # A list of ghost positions
```

After getting particle, I called getPositionDistribution method to update particle. Then we got sample from all updated particles.

```
for i in range(self.numGhosts):
    newParticle[i] = (self.getPositionDistribution(gameState, oldParticle, i, self.ghostAgents[i])).sample()
```

To update weights, I again iterated over particles which is list of ghost beliefs. Then to update 1 particle, we go over each ghost observations by using getObservationProb method.

```
w_distribution = DiscreteDistribution()
for p in self.particles:
    w = 1
    for i in range(self.numGhosts):
        w *= self.getObservationProb(observation[i], gameState.getPacmanPosition(), p[i],
                                     self.getJailPosition(i))
    w_distribution[p] += w
```