- **Show your autograder results and describe the implementation details:**
  - Q1.

```
Question q1
===========
*** PASS: test_cases/q1/1-small-board.test
*** PASS: test_cases/q1/2-long-bottom.test
*** PASS: test_cases/q1/3-wide-inverted.test

### Question q1: 2/2 ###


Finished at 19:45:27

Provisional grades
==================
Question q1: 2/2
------------------
Total: 2/2
```

1. 變數和邊的定義：
   - Variables 包含了所有的節點，也就是 Pacman, Ghost1, 2 的位置和觀測點
   - Edges 定義了節點之間的依賴關係，例如：從 Pacman 的位置到其觀測點的距離，從 Ghost 到觀測點的對應觀測距離。
2. 變數值域的設置：
   - 對於位置變數（Pacman and Ghost），他們的可能值是所有非障礙物的格子位置，形成一個二維座標的列表。
   - 對於觀測變數，因為觀測到的是帶 Noise 的 Manhattan Distance，其可能的值是從 0 到最大可能距離（這邊設定的是：$x + y - 1 + Max\_Noise$）

最後利用上述定義的變數、邊和值域來構造一個空的 Bayes Net，其主要用途是在遊戲的後續處理中，利用這種概率模型來推斷 Pacman, Ghost 的位置，實現更智能的遊戲策略和決策。

- Q2.

```
Question q2
===========
*** PASS: test_cases/q2/1-product-rule.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/2-product-rule-extended.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/3-disjoint-right.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/4-common-right.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/5-grade-join.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q2/6-product-rule-nonsingleton-var.test
***     Executed FactorEqualityTest

### Question q2: 3/3 ###


Finished at 20:33:23

Provisional grades
==================
Question q2: 3/3
------------------
Total: 3/3
```

1. 提取變數域字典：
   假設所有因子都來自同一 Bayes Net，因此他們的變數域字典是相同的，這裡從因子列表中的第一個因子提取變數域字典。
2. 確定 unconditioned variables, conditioned variables:
   遍歷每個因子，收集所有因子的 unconditioned variables and conditioned variables，然後從 conditioned variables set 中移除那些也被當作 unconditioned variable 的 variables，以確保變數的正確分類。
3. 創建新的合併因子：
   使用更新後的 unconditioned variable set, conditioned variable set，以及從因子中獲得的變數域字典，創建一個新的因子。
4. 計算新因子的概率表：
   為新創建的因子計算概率，這一步驟包括遍歷新因子的所有可能賦值字典，對於每一個賦值，計算所有原始因子在該賦值下的概率乘積，並將這個乘積設置為新因子在該賦值下的概率。

最後這個函數返回合併後的新因子，該因子包含了所有給定因子的資訊以及他們的概率乘積。總結來說，joinFactors 函數透過合併多個因子的變數和概率表來構造一個包含更廣泛資訊的新因子，這對於 Bayes Net 中進行概率推理非常重要。

- Q3.

```
Question q3
===========
*** PASS: test_cases/q3/1-simple-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/2-simple-eliminate-extended.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/3-eliminate-conditioned.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/4-grade-eliminate.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/5-simple-eliminate-nonsingleton-var.test
***     Executed FactorEqualityTest
*** PASS: test_cases/q3/6-simple-eliminate-int.test
***     Executed FactorEqualityTest

### Question q3: 2/2 ###


Finished at 21:02:36

Provisional grades
==================
Question q3: 2/2
------------------
Total: 2/2
```

1. 變量集處理：
   從因子中獲取 unconditioned variable set, conditioned variable set，將要消除的變量從 unconditioned variable set 中移除，conditioned variable set 保持不變。
2. 創建新的因子：
   使用更新後的 unconditioned variable set, original conditioned set，以及從原始因子中繼承的變量域字典，創建一個新的因子。
3. 計算新因子的概率表：
   遍歷新因子的所有可能賦值，對於每個賦值，考慮所有消除變量可能的值，對應的將原始因子中與這些賦值匹配的概率進行累加，從而計算出在消除該變量後的概率。累加完成後，將計算出的概率值設置到新因子對應的賦值中。
4. 返回結果：
   函數最終返回新創建的因子，該因子不再包含被消除的變量，並具有基於原始因子概率的合計概率。

總的來說，eliminate 函數實現了對因子中某個 unconditioned variable 的消除操作，通過合併概率來組合所有可能的情境，從而使得因子的表示更為簡潔，也便於進一步的計算和推理，這是構建和操作 Bayes Net 中常見的一種技術。

- Q4.

```
### Question q4: 2/2 ###


Finished at 20:32:00

Provisional grades
==================
Question q4: 2/2
------------------
Total: 2/2
```

1. 設定和初始化：
   函數首先確定變數消除的順序，如果未指定則自動生成。接著從 Bayes Net 中獲取與證據相關的所有條件概率表（CPTs）
2. 變數消除過程：
   按照指定的消除順序，對每個變數進行操作。首先使用 joinFactorsByVariable 將涉及該變數的所有因子合併為一個因子，然後檢查是否可以對這個合併的因子進行消除操作。如果合併因子的 unconditioned variable 多於一個，則使用 eliminate 函數來消除該變數。
3. 因子更新與合併：
   消除操作後的因子將被加入到當前因子列表中，這樣逐步更新因子列表。最後將所有剩餘的因子合併成一個因子。
4. 結果的 Normalization
   最後一步是 normalized 後的因子，該因子表示查詢變數在給定證據的條件下的概率分佈。

總結來說，inferenceByVariableElimination 函數透過連接和消除變量的交替過程來處理和簡化 Bayes Net，最終得到一個對給定證據條件畫的查詢變數的概率分佈。這種方法是 Bayes Net 推理中的一種常用技術，特別適用於處理含有大量變量的複雜網路。

- Q5.

```
Question q5
===========
*** PASS: test_cases/q5/1-DiscreteDist.test
***     PASS
*** PASS: test_cases/q5/1-DiscreteDist-a1.test
***     PASS
*** PASS: test_cases/q5/1-ObsProb.test
***     PASS

### Question q5: 1/1 ###


Finished at 20:46:33

Provisional grades
==================
Question q5: 1/1
------------------
Total: 1/1
```

  - Q5a.

    Normalize: 將離散分佈的值正規化，使總和為 1，從而可以表示為概率分佈

    Sample: 用於根據已經 normalized 的離散概率分佈抽樣，返回一個隨機抽取的鍵，抽取的概率與鍵對應的值成正比。

  - Q5b.

    getObservation 函數是用來計算在特定情境下的觀測機率，即在已知 Pacman 和 Ghost 的位置的情況下，獲得某個 Noise Distance 的概率。以下是函數的具體實現細節：

    1. 檢查鬼是否在監獄中：

       如果鬼的位置與監獄的位置相同，則認為鬼被囚禁。在這種情況下，如果沒有觀測到噪聲距離（即 noisyDistance 為 None），則返回概率 1（表示這是預期的觀測結果）。如果有觀測到噪聲距離，則返回概率 0，因為囚禁的鬼不應該有任何距離觀測。

    2. 計算實際距離和噪聲觀測的概率：

       如果鬼不在監獄中且存在噪聲距離的觀測，首先計算 Pacman 和 Ghost 之間的 manhattan distance。然後，使用這個實際距離和觀測到的噪聲距離來計算概率，這通過調用 busters.getObservationProbability 函數實現。

- Q6.

```
Question q6
===========
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/1-ExactUpdate.test
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/2-ExactUpdate.test
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/3-ExactUpdate.test
*** q6) Exact inference stationary pacman observe test: 0 inference errors.
*** PASS: test_cases/q6/4-ExactUpdate.test

### Question q6: 2/2 ###


Finished at 21:05:02

Provisional grades
==================
Question q6: 2/2
------------------
Total: 2/2
```

observeUpdate 函數是用於更新根據觀測結果修整對鬼位置的信仰（概率分佈）。這種更新考慮了 Pacman 當前的位置和對鬼的噪聲 manhattan distance 觀測。以下是詳細的實現步驟：

1. 獲取 Pacman 位置
2. 初始化新的信仰分佈：
   創建一個新的 DiscreteDistribution 對象來存儲更新後的信仰。
3. 更新每個可能的鬼的位置的信仰：
   - 遍歷所有可能的鬼的位置（包括監獄位置）。對於每個位置，計算給定當前觀測值和 Pacman 位置時，該位置的鬼出現的觀測概率。這是透過調用 getObservationProb 函數來實現的。
   - 使用觀測概率來更新鬼位置的信仰值。新的信仰值是原始信仰值與觀測概率的乘積。
4. 正規化信仰分佈：
   更新完所有鬼位置的信仰後，需要正規化這個分佈，確保所有概率值之和為 1。這是透過調用 normalize 函數來實現的。
5. 更新內部信仰狀態：
   將類的 beliefs 屬性更新為新的、已正規化的信仰分佈。

透過上述步驟，這個函數有效地根據新的觀測數據調整了對鬼位置的估計，使之更加精確。這種方法允許動態地根據每次新的觀測來更新位置的概率分布，非常適合於動態變化的遊戲環境中進行實時決策和推理。

- Q7.

```
Question q7
===========
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/1-ExactPredict.test
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/2-ExactPredict.test
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/3-ExactPredict.test
*** q7) Exact inference elapseTime test: 0 inference errors.
*** PASS: test_cases/q7/4-ExactPredict.test

### Question q7: 2/2 ###


Finished at 21:16:44

Provisional grades
==================
Question q7: 2/2
------------------
Total: 2/2
```

elapseTime 函數是用來更新鬼的位置信仰（概率分布），以響應遊戲時間的推移。此函數基於鬼的移動轉移模型，該模型可能依賴於吃豆人的當前位置，但由於吃豆人位置已知，這不構成問題。以下是具體的實現步驟：

1. 初始化新的信仰分佈
2. 獲取 Pacman 位置
3. 處理每個可能的鬼位置：
   - 遍歷所有可能的鬼的當前位置（即 self.allPositions 中的每個位置）。對於每個位置：
     i. 調用 getPositionDistribution 函數獲取從該位置到其他所有位置的過度概率分佈。這個分佈反映了鬼可能移動到新位置的概率。
     ii. 遍歷這個過度概率分佈，對於分佈中的每個新位置及其概率：
       - 將從舊位置到新位置的概率與舊位置的當前信仰值（即 self.beliefs[oldPos]）相乘，然後累加到新位置的信仰值中。
4. 更新信仰狀態

通過這個函數，每當遊戲時間推進，就會根據鬼可能的移動行為更新其位置的概率分布，進而能夠更好地預測鬼的未來位置。這個過程考慮了鬼的動態移動，使得模型更能適應遊戲的變化，從而提高跟蹤和預測的準確性。

- Q8.

```
Question q8
===========
*** q8) Exact inference full test: 0 inference errors.
*** PASS: test_cases/q8/1-ExactFull.test
*** q8) Exact inference full test: 0 inference errors.
*** PASS: test_cases/q8/2-ExactFull.test
ExactInference
[Distancer]: Switching to maze distances
Average Score: 763.3
Scores:        778, 769, 759, 761, 776, 761, 758, 753, 763, 755
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** Won 10 out of 10 games. Average score: 763.300000 ***
*** smallHunt) Games won on q8 with score above 700: 10/10
*** PASS: test_cases/q8/3-gameScoreTest.test

### Question q8: 1/1 ###


Finished at 21:25:52

Provisional grades
==================
Question q8: 1/1
------------------
Total: 1/1
```

獲取 Pacman 當前位置和合法行動、獲取活著的鬼的位置分佈、計算最可能的鬼的位置、計算最近的鬼的位置、評估各個行動帶來的新位置距離後選擇最佳行動

- Q9.

```
Question q9
===========
*** q9) Particle filter initialization test: 0 inference errors.
*** PASS: test_cases/q9/1-ParticleInit.test
*** q9) numParticles initialization test: 0 inference errors.
*** PASS: test_cases/q9/2-ParticleInit.test

### Question q9: 1/1 ###


Finished at 21:30:52

Provisional grades
==================
Question q9: 1/1
------------------
Total: 1/1
```

這兩個函數 initializeUniformly 和 getBeliefDistribution 是用於初始化和管理粒子濾波器中的粒子，以對鬼的位置進行概率估計。以下是兩個函數的實現細節：

initializeUniformly 這個函數的目的是均勻地分佈粒子到所有合法的位置，以確保初始的概率分佈是均勻的。

getBeliefDistribution 函數

這個函數的目的是將粒子列表轉換成一個信仰分佈（概率分佈），反映基於所有證據和時間推移後的鬼位置的估計。

- Q10. （和第 6 題一樣都是 observeUpdate，但使用的是粒子濾波）

```
Question q10
============
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/1-ParticleUpdate.test
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/2-ParticleUpdate.test
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/3-ParticleUpdate.test
*** q10) Particle filter observe test: 0 inference errors.
*** PASS: test_cases/q10/4-ParticleUpdate.test
*** q10) successfully handled all weights = 0
*** PASS: test_cases/q10/5-ParticleUpdate.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 175.5
Scores:        188, 192, 198, 186, 184, 193, 144, 172, 197, 101
Win Rate:      10/10 (1.00)
Record:        Win, Win, Win, Win, Win, Win, Win, Win, Win, Win
*** Won 10 out of 10 games. Average score: 175.500000 ***
*** oneHunt) Games won on q10 with score above 100: 10/10
*** PASS: test_cases/q10/6-ParticleUpdate.test

### Question q10: 2/2 ###


Finished at 21:35:30

Provisional grades
==================
Question q10: 2/2
------------------
Total: 2/2
```

這個 observeUpdate 函數是用於根據觀測到的噪聲 manhattan distance 和 Pacman 的位置更新對鬼位置的信仰分布。這裡使用粒子濾波器進行更新，考慮到可能所有粒子都會收到零權重的特殊情況。以下是函數的實現細節：

✓ 更新粒子權重

1. 初始化新權重分佈
2. 更新每個粒子的權重
   - i. 遍歷每個粒子，對於每個粒子，計算給定觀測值和 Pacman 位置時該粒子代表的位置的觀測概率。這是通過調用 getObservationProb 函數來實現。
   - ii. 該粒子在新權重分佈中的值，增加相應的觀測概率。

✓ 處理所有粒子權重為 0 的情況

3. 檢查所有粒子權重是否為 0：
   - i. 如果更新後的權重分佈的總和為 0（即所有粒子權重都為 0），則調用 initializeUniformly 函數重新均勻初始化粒子。
   - ii. 在重新初始化後，將所有粒子的權重設為均一值。

✓ 根據權重重新抽樣粒子

4. 重新抽樣粒子：
   i. 基於更新的權重分佈重新抽樣等數量的粒子。這是透過多次調用 DiscreteDistribution.sample 方法來實現，每次根據權重分佈隨機選擇一個粒子。
   ii. 更新粒子列表，使其反映重新抽樣後的結果。

通過上述步驟，這個函數有效地根據新的觀測數據調整了對鬼位置的估計，並且能夠處理所有粒子權重變為零的特殊情況，確保粒子濾波器不會因為權重損失而失效。這種方法使得模型可以適應於觀測數據帶來的信息變化，從而提供對動態環境中的位置估計更穩健的更新。

- Q11.

```
Question q11
============
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/1-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/2-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/3-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/4-ParticlePredict.test
*** q11) Particle filter full test: 0 inference errors.
*** PASS: test_cases/q11/5-ParticlePredict.test
ParticleFilter
[Distancer]: Switching to maze distances
Average Score: 364.0
Scores:        380, 361, 361, 383, 335
Win Rate:      5/5 (1.00)
Record:        Win, Win, Win, Win, Win
*** Won 5 out of 5 games. Average score: 364.000000 ***
*** smallHunt) Games won on q11 with score above 300: 5/5
*** PASS: test_cases/q11/6-ParticlePredict.test

### Question q11: 2/2 ###


Finished at 21:47:06

Provisional grades
==================
Question q11: 2/2
------------------
Total: 2/2
```

這個 elapseTime 函數是用於模擬時間流逝下粒子（代表鬼的可能位置）的狀態變化，這是在粒子濾波器的框架下進行的。以下是具體的實現步驟和細節：

✓ 功能概述

函數的目的是根據遊戲的當前狀態和每個粒子的當前狀態來更新或預測每個粒子的下一個狀態。這個過程涉及到根據粒子代表的

鬼的可能位置和遊戲狀態來預測鬼的下一步行動。

- ✓ 實現步驟
  1. 初始化新粒子列表
     創建一個空列表 newParticles，用於存儲每個粒子的新位置。
  2. 遍歷現有粒子：
     對於 self.particles 中的每個粒子（代表一個可能的鬼的位置），執行以下步驟：
  3. 獲取轉移模型：
     調用 getPositionDistribution 方法，傳入當前遊戲狀態和粒子當前的位置（oldParticle），獲取該粒子根據遊戲規則可能移動到的新位置的概率分佈。這個分佈表示了從當前位置出發，鬼可能的移動方向和位置。
  4. 從概率分佈中抽樣新位置：
     使用從 getPositionDistribution 獲得的概率分布，調用其 sample 方法隨機選擇一個新位置，作為這個粒子在下一時間步的位置。
  5. 更新粒子列表：
     i. 將抽樣得到的新位置添加到 newParticles 列表中。
     ii. 更新 self.particles 為 newParticles，這樣粒子列表就包含了所有粒子根據預測模型更新後的位置。
- ✓ 功能和應用

  這個函數通過預測每個粒子的下一個位置，允許粒子濾波器模型適應鬼的動態行為，從而更準確地跟蹤鬼的位置。在策略遊戲如吃豆人這樣的設定中，能夠有效地預見對手的可能行動是非常重要的，此函數提供了一種模擬對手行動的機制。