Question 1

a.

i run ***python lwUmbrella.py 1000 10 1111100000***

```
P(R_10 = T|u_1:10) = 0.0510
P(R_10 = F|u_1:10) = 0.9490

P(R_10 = T|u_1:10) = 0.0454
P(R_10 = F|u_1:10) = 0.9546

P(R_10 = T|u_1:10) = 0.0503
P(R_10 = F|u_1:10) = 0.9497

P(R_10 = T|u_1:10) = 0.0819
P(R_10 = F|u_1:10) = 0.9181

P(R_10 = T|u_1:10) = 0.0601
P(R_10 = F|u_1:10) = 0.9399

P(R_10 = T|u_1:10) = 0.0581
P(R_10 = F|u_1:10) = 0.9419

P(R_10 = T|u_1:10) = 0.0863
P(R_10 = F|u_1:10) = 0.9137

P(R_10 = T|u_1:10) = 0.0355
P(R_10 = F|u_1:10) = 0.9645

P(R_10 = T|u_1:10) = 0.0641
P(R_10 = F|u_1:10) = 0.9359

P(R_10 = T|u_1:10) = 0.0496
P(R_10 = F|u_1:10) = 0.9504

After running 10 time, the varience of estimates is 0.00023
```

ii run ***python lwUmbrella.py 1000 10 0000000111***

```
P(R_10 = T|u_1:10) = 0.9148
P(R_10 = F|u_1:10) = 0.0852

P(R_10 = T|u_1:10) = 0.8302
```

```
P(R_10 = F|u_1:10) = 0.1698

P(R_10 = T|u_1:10) = 0.8338
P(R_10 = F|u_1:10) = 0.1662

P(R_10 = T|u_1:10) = 0.8729
P(R_10 = F|u_1:10) = 0.1271

P(R_10 = T|u_1:10) = 0.9275
P(R_10 = F|u_1:10) = 0.0725

P(R_10 = T|u_1:10) = 0.8912
P(R_10 = F|u_1:10) = 0.1088

P(R_10 = T|u_1:10) = 0.8253
P(R_10 = F|u_1:10) = 0.1747

P(R_10 = T|u_1:10) = 0.8701
P(R_10 = F|u_1:10) = 0.1299

P(R_10 = T|u_1:10) = 0.8856
P(R_10 = F|u_1:10) = 0.1144

P(R_10 = T|u_1:10) = 0.9023
P(R_10 = F|u_1:10) = 0.0977

After running 10 time, the varience of estimates is 0.00117
```

iii run ***python lwUmbrella.py 1000 10 0101010101***

```
P(R_10 = T|u_1:10) = 0.7453
P(R_10 = F|u_1:10) = 0.2547

P(R_10 = T|u_1:10) = 0.6556
P(R_10 = F|u_1:10) = 0.3444

P(R_10 = T|u_1:10) = 0.5090
P(R_10 = F|u_1:10) = 0.4910

P(R_10 = T|u_1:10) = 0.6519
P(R_10 = F|u_1:10) = 0.3481

P(R_10 = T|u_1:10) = 0.6152
P(R_10 = F|u_1:10) = 0.3848

P(R_10 = T|u_1:10) = 0.7164
P(R_10 = F|u_1:10) = 0.2836
```

```
P(R_10 = T|u_1:10) = 0.7323
P(R_10 = F|u_1:10) = 0.2677

P(R_10 = T|u_1:10) = 0.7094
P(R_10 = F|u_1:10) = 0.2906

P(R_10 = T|u_1:10) = 0.7283
P(R_10 = F|u_1:10) = 0.2717

P(R_10 = T|u_1:10) = 0.6656
P(R_10 = F|u_1:10) = 0.3344

After running 10 time, the varience of estimates is 0.00461
```

b.

i run ***python pfUmbrella.py 1000 10 1111100000***

```
P(R_10 = T|u_1:10) = 0.0580
P(R_10 = F|u_1:10) = 0.9420

P(R_10 = T|u_1:10) = 0.0510
P(R_10 = F|u_1:10) = 0.9490

P(R_10 = T|u_1:10) = 0.0640
P(R_10 = F|u_1:10) = 0.9360

P(R_10 = T|u_1:10) = 0.0570
P(R_10 = F|u_1:10) = 0.9430

P(R_10 = T|u_1:10) = 0.0440
P(R_10 = F|u_1:10) = 0.9560

P(R_10 = T|u_1:10) = 0.0570
P(R_10 = F|u_1:10) = 0.9430

P(R_10 = T|u_1:10) = 0.0470
P(R_10 = F|u_1:10) = 0.9530

P(R_10 = T|u_1:10) = 0.0700
P(R_10 = F|u_1:10) = 0.9300

P(R_10 = T|u_1:10) = 0.0570
P(R_10 = F|u_1:10) = 0.9430

P(R_10 = T|u_1:10) = 0.0670
```

```
P(R_10 = F|u_1:10) = 0.9330

After running 10 time, the varience of estimates is 0.00006
```

ii run ***python pfUmbrella.py 1000 10 0000000111***

```
P(R_10 = T|u_1:10) = 0.8990
P(R_10 = F|u_1:10) = 0.1010

P(R_10 = T|u_1:10) = 0.8950
P(R_10 = F|u_1:10) = 0.1050

P(R_10 = T|u_1:10) = 0.8820
P(R_10 = F|u_1:10) = 0.1180

P(R_10 = T|u_1:10) = 0.8660
P(R_10 = F|u_1:10) = 0.1340

P(R_10 = T|u_1:10) = 0.8910
P(R_10 = F|u_1:10) = 0.1090

P(R_10 = T|u_1:10) = 0.8790
P(R_10 = F|u_1:10) = 0.1210

P(R_10 = T|u_1:10) = 0.8960
P(R_10 = F|u_1:10) = 0.1040

P(R_10 = T|u_1:10) = 0.9050
P(R_10 = F|u_1:10) = 0.0950

P(R_10 = T|u_1:10) = 0.9010
P(R_10 = F|u_1:10) = 0.0990

P(R_10 = T|u_1:10) = 0.8950
P(R_10 = F|u_1:10) = 0.1050

After running 10 time, the varience of estimates is 0.00013
```

iii run ***python pfUmbrella.py 1000 10 0101010101***

```
P(R_10 = T|u_1:10) = 0.7540
P(R_10 = F|u_1:10) = 0.2460

P(R_10 = T|u_1:10) = 0.7160
P(R_10 = F|u_1:10) = 0.2840

P(R_10 = T|u_1:10) = 0.6940
P(R_10 = F|u_1:10) = 0.3060
```

```
P(R_10 = T|u_1:10) = 0.6920
P(R_10 = F|u_1:10) = 0.3080

P(R_10 = T|u_1:10) = 0.7010
P(R_10 = F|u_1:10) = 0.2990

P(R_10 = T|u_1:10) = 0.7120
P(R_10 = F|u_1:10) = 0.2880

P(R_10 = T|u_1:10) = 0.6990
P(R_10 = F|u_1:10) = 0.3010

P(R_10 = T|u_1:10) = 0.7160
P(R_10 = F|u_1:10) = 0.2840

P(R_10 = T|u_1:10) = 0.7470
P(R_10 = F|u_1:10) = 0.2530

P(R_10 = T|u_1:10) = 0.7260
P(R_10 = F|u_1:10) = 0.2740
```

`After running 10 time, the varience of estimates is 0.00041`

c. I have written a code to exactly calculate in DBN.

i run ***python exactDBN.py 10 1111100000***

```
P(R_10 = T|u_1:10) = 0.0562
P(R_10 = F|u_1:10) = 0.9438
```

ii run ***python exactDBN.py 10 0000000111***

```
P(R_10 = T|u_1:10) = 0.8902
P(R_10 = F|u_1:10) = 0.1098
```

iii run ***python exactDBN.py 10 0101010101***

```
P(R_10 = T|u_1:10) = 0.7171
P(R_10 = F|u_1:10) = 0.2829
```

compare the results with what we got in (a) and (b), it is safe to conclude that Particle Filtering is more accurate and stable than Likelihood Weight method, since it is more close to exact result and its variance is smaller.

Question 2

a.

- run ***python mdpVI.py -2***

```
s = (1, 1) : U[s] = -10.8153 , a = r
s = (1, 2) : U[s] = -9.5425 , a = u
s = (1, 3) : U[s] = -7.0425 , a = r
```

```
s = (2, 1) : U[s] = -8.4744 , a = r
s = (2, 3) : U[s] = -4.2300 , a = r
s = (3, 1) : U[s] = -5.9744 , a = r
s = (3, 2) : U[s] = -3.5704 , a = r
s = (3, 3) : U[s] = -1.7300 , a = r
s = (4, 1) : U[s] = -3.7749 , a = u
s = (4, 2) : U[s] = -1.0000 , a = N
s = (4, 3) : U[s] = 1.0000 , a = N
```

- run ***python mdpVI.py -0.2***

```
s = (1, 1) : U[s] = -0.3273 , a = u
s = (1, 2) : U[s] = -0.0826 , a = u
s = (1, 3) : U[s] = 0.1674 , a = r
s = (2, 1) : U[s] = -0.2848 , a = r
s = (2, 3) : U[s] = 0.4486 , a = r
s = (3, 1) : U[s] = -0.0348 , a = u
s = (3, 2) : U[s] = 0.2877 , a = u
s = (3, 3) : U[s] = 0.6986 , a = r
s = (4, 1) : U[s] = -0.3642 , a = l
s = (4, 2) : U[s] = -1.0000 , a = N
s = (4, 3) : U[s] = 1.0000 , a = N
```

- run ***python mdpVI.py -0.01***

```
s = (1, 1) : U[s] = 0.9232 , a = u
s = (1, 2) : U[s] = 0.9372 , a = u
s = (1, 3) : U[s] = 0.9497 , a = r
s = (2, 1) : U[s] = 0.9107 , a = l
s = (2, 3) : U[s] = 0.9638 , a = r
s = (3, 1) : U[s] = 0.8969 , a = l
s = (3, 2) : U[s] = 0.8866 , a = l
s = (3, 3) : U[s] = 0.9763 , a = r
s = (4, 1) : U[s] = 0.7969 , a = d
s = (4, 2) : U[s] = -1.0000 , a = N
s = (4, 3) : U[s] = 1.0000 , a = N
```

b.

- run ***python mdpPI.py -2***

```
s = (1, 1) : U[s] = -10.8153 , a = r
s = (1, 2) : U[s] = -9.5425 , a = u
s = (1, 3) : U[s] = -7.0425 , a = r
s = (2, 1) : U[s] = -8.4744 , a = r
s = (2, 3) : U[s] = -4.2300 , a = r
s = (3, 1) : U[s] = -5.9744 , a = r
s = (3, 2) : U[s] = -3.5704 , a = r
s = (3, 3) : U[s] = -1.7300 , a = r
s = (4, 1) : U[s] = -3.7749 , a = u
s = (4, 2) : U[s] = -1.0000 , a = N
```

```
s = (4, 3) : U[s] = 1.0000 , a = N
```

- run ***python mdpPI.py -0.2***

```
s = (1, 1) : U[s] = -0.3273 , a = u
s = (1, 2) : U[s] = -0.0826 , a = u
s = (1, 3) : U[s] = 0.1674 , a = r
s = (2, 1) : U[s] = -0.2848 , a = r
s = (2, 3) : U[s] = 0.4486 , a = r
s = (3, 1) : U[s] = -0.0348 , a = u
s = (3, 2) : U[s] = 0.2877 , a = u
s = (3, 3) : U[s] = 0.6986 , a = r
s = (4, 1) : U[s] = -0.3642 , a = l
s = (4, 2) : U[s] = -1.0000 , a = N
s = (4, 3) : U[s] = 1.0000 , a = N
```

- run ***python mdpPI.py -0.01***

```
s = (1, 1) : U[s] = 0.9232 , a = u
s = (1, 2) : U[s] = 0.9372 , a = u
s = (1, 3) : U[s] = 0.9497 , a = r
s = (2, 1) : U[s] = 0.9107 , a = l
s = (2, 3) : U[s] = 0.9638 , a = r
s = (3, 1) : U[s] = 0.8969 , a = l
s = (3, 2) : U[s] = 0.8866 , a = l
s = (3, 3) : U[s] = 0.9763 , a = r
s = (4, 1) : U[s] = 0.7969 , a = d
s = (4, 2) : U[s] = -1.0000 , a = N
s = (4, 3) : U[s] = 1.0000 , a = N
```

It can be seen that mdpPI.py and mdpVI.py give exactly the same solution.

Question 3

Using the idea of Dynamical Programming, we can efficiently compute $argmax_{x_1,\cdots,x_{100}} f(x)$ let

$$V(j, x_1, x_{j+1}, x_{j+2}) = \max_{x_2,x_3,\cdots,x_j} \phi_1(x_1, x_2, x_3)\phi_1 \cdots \phi_{j-1}\phi_j(x_j, x_{j+1}, x_{j+2}) \tag{1}$$

then we have

$$V(j+1, x_1, x_{j+2}, x_{j+3}) = \max_{x_{j+1}} V(j, x_1, x_{j+1}, x_{j+2})\phi_{j+1}(x_{j+1}, x_{j+2}, x_{j+3}) \tag{2}$$

so

$$f(x) = \max_{x_1,x_{99},x_{100}} \phi_0(x_1, x_{100})V(98, x_1, x_{99}, x_{100}) \tag{3}$$

since we are asked to compute $argmax_{x_1,\cdots,x_{100}} f(x)$, we just need to record where V comes from, to be concrete, let

$$record(j+1, x_1, x_{j+2}, x_{j+3}) = argmax_{x_{j+1}} V(j, x_1, x_{j+1}, x_{j+2})\phi_{j+1}(x_{j+1}, x_{j+2}, x_{j+3}) \tag{4}$$

thus we can construct the set $\{x_1, x_2, \cdots, x_{100}\}$ from the array record.
Anaysis:
(2) requires $O(1)$, there are 100×2^3 states, thus total time complexity would be $O(100\times 2^3)$

Question 4

a. proof:
We have

$$\sum_k P(E_j = e_{jk}|E)EU(\alpha|E, E_j = e_{jk}) = EU(\alpha|E)$$

By definition,

$$EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) \geq EU(\alpha|E, E_j = e_{jk})$$

Combine those facts, we have

$$\begin{aligned}
\mathrm{VPI}_E(E_j) &= \left(\sum_k P(E_j = e_{jk}|E)EU(\alpha_{e_{jk}}|E, E_j = e_{jk})\right) - EU(\alpha|E) \\
&\geq \left(\sum_k P(E_j = e_{jk}|E)EU(\alpha|E, E_j = e_{jk})\right) - EU(\alpha|E) \\
&= 0
\end{aligned}$$

b. To simplify the notation, denote $\mathrm{VPI}_E(E_j) = V_E(E_j) - EU(\alpha|E)$, thus $V_E(E_j) = \sum_{e_{jn}} P(E_j = e_{jn}|E)EU(\alpha_{jn}|E, E_j = e_{jn})$, by this definition, $EU(\alpha|E) = V_E()$.
Now,

$$\mathrm{VPI}_E(E_j, E_k) = V_E(E_j, E_k) - V_E() \tag{5}$$

Note

$$V_E(E_j) = \sum_{e_{jn}} P(E_j = e_{jn}|E)EU(\alpha_{jn}|E, E_j = e_{jn}) = \sum_{e_{jn}} P(E_j = e_{jn}|E)V_{E,E_j=e_{jn}}()$$

similarly, we have

$$V_E(E_j, E_k) = \sum_{e_{jn}} P(E_j = e_{jn}|E)V_{E,E_j=e_{jn}}(E_k)$$

Note that

$$\begin{aligned}
(5) &= \Big(V_E(E_j, E_k) - V_E(E_j)\Big) + \Big(V_E(E_j) - V_E()\Big) \\
&= \sum_{e_{jn}} P(E_j = e_{jn}|E)\Big(V_{E,E_j=e_{jn}}(E_k) - V_{E,E_j=e_{jn}}()\Big) + \mathrm{VPI}_E(E_j) \\
&= \sum_{e_{jn}} P(E_j = e_{jn}|E)\mathrm{VPI}_{E,E_j=e_{jn}}(E_k) + \mathrm{VPI}_E(E_j) \\
&= \mathrm{VPI}_{E,E_j}(E_k) + \mathrm{VPI}_E(E_j)
\end{aligned}$$

since we have

$$(5) = \Big(V_E(E_k, E_j) - V_E(E_k)\Big) + \Big(V_E(E_k) - V_E()\Big)$$

We can similarly prove

$$(5) = \mathrm{VPI}_{E,E_k}(E_j) + \mathrm{VPI}_E(E_k)$$

Thus we have got what we want.