Questions for Grading level C

Question 1:
Pi = [0.5, 0.5]
A = [[0.5, 0.5], [0.5, 0.5]]
B = [[0.9, 0.1], [0.5, 0.5]]

Question 2:
Getting the following state probability
[0, 1] * [[0.5, 0.5], [0.5, 0.5]] = [0.5, 0.5]

Question 3:
Getting the probabilities of the emitted observation of that state
[0.5, 0.5] * [[0.9, 0.1], [0.5, 0.5]] = [0.7, 0.3]

Question 4:
Because when we have the condition that the state at time t is $X_t = x_i$, the observation at time t O_t is conditionally independent of $O_{1:t-1}$. (Current observations are independent from previous once)

Question 5:
Delte is used in the Viterbi algorithm and contain the maximum probability to be in each state at a given time, while delta^idx contains the respective previous states for each of deltas probabilities.
The matrix δ(delta) and δ^{idx} stores the same number of parameters as the number of states.

Question 6:
Di-gamma function is defined as the probability of the hidden states $X_t = x_i$ and $X_{t+1} = x_j$, given the whole observation sequence. (Probabiulity of being in X_i given the observations and the next state being X_j) Based on Bayes theorem:

$$P\left(X_t = x_i,\ X_{t+1} = x_j \middle| O_{1:T}\right) = \frac{P(X_t = x_i,\ \ X_{t+1} = x_j,\ O_{1:T})}{P(O_{1:T})}$$

In which $P\left(O_{1:T}\right) = sum(\alpha_T)$
So we have to divide by sum(a_T), because we use Bayes and the corresponding parameter(normalisation factor) in this scenario is given by sum over alphas.

Question 7:
Does the algorithm converge?
For 1000 observations the algorithm exited the max iteration of 500 and therefore does not converge.
Result:

```
 A = [[0.701783, 0.009971, 0.288246],
      [0.099539, 0.812977, 0.087484],
      [0.188647, 0.303974, 0.507379]]

B = [[0.685501, 0.224179, 0.076645, 0.013675],
     [0.070764, 0.412268, 0.279866, 0.237101],
     [0.0,      0.000392, 0.354053, 0.645555]]
```

For 10000 observations the algorithm exited the max iteration of 500 and therefore does not converge.

```
A = [[0.695734, 0.042127, 0.262138],
     [0.118171, 0.743221, 0.138607],
     [0.152364, 0.253569, 0.594067]]
```

```
B = [[0.708497, 0.186341, 0.103398, 0.001764],
     [0.099432, 0.423989, 0.311417, 0.165162],
     [0.03284,   0.172778, 0.190184, 0.604198]]
```

How many observations do you need for the algorithm to converge?
More observations can help the algorithm to detect the underlying structure of the data. But in the end it also comes down to the general parametrisation, like max-iterations and so on.
As soon as a certain threshold of data, to comprehensively describe the underlying structure/ phenomena, for training is passed we can achieve convergence.

How can you define convergence?
If the training process would be visualised as a U shaped curve, convergence of the algorithm would look like that with each iteration the estimate gets closer to the valley(local min) of the curve(gradient decent). At some point its step size gets just too big and it starts jumping from one side of the pit to the other always over- or under- estimating the perfect result by a bit.

Question 8:

```
A = [[0.625093, 0.240202, 0.134706],
     [0.179883, 0.563137, 0.25698 ],
     [0.181709, 0.006437, 0.811854]]

B =  [[0.709915, 0.2261    , 0.063985, 0.0       ],
      [0.057846, 0.0       , 0.337204, 0.60495  ],
      [3.3e-05.  , 0.358089, 0.312826, 0.329052]]
```

WHAT IS THE PROBLEM WHEN IT COMES TO ESTIMATING THE DISTANCE BETWEEN THESE MATRICES? HOW CAN YOU SOLVE THESE ISSUES?
We can use for example the Euclidian distance to calculate the distance between the matrices, but these distances can be from matrix elements which might vary a lot in the magnitude of their values. For example if we compare a matrix similar to A except of entry A(1,2) = 0.15 instead of 0.05 and one that differs in position A(2,2) = 0.9 instead of 0.8.
Both values differ by 0.1 having the same euclidian distance from A, but in the first case that is a change in the order of magnitude one. Therefore it should be less similar to A than the second option. The distance function at use is essential here and might need to take in to account more than just absolut values. A possible solution might be normalisation to keep the change uniform. Or instead use relative errors instead of absolut to calculate the distances.

QUESTION 9:
RESULT:
A(5 5) = 0.267428 0.109426 0.331097 0.208661 0.083389 0.058384 0.541403 0.080251 0.186481 0.133481 0.400067 0.075553 0.273424 0.125791 0.125165 0.201641 0.170677 0.198819 0.237586 0.191278 0.014699 0.226393 0.011619 0.05291 0.694379
B(5, 4) = 0.064055 0.449324 0.313053 0.173569 0.020986 0.149501 0.195885 0.633628 0.108159 0.402409 0.378844 0.110588 0.126048 0.359881 0.181456 0.332616 0.708436 0.18388 0.104413 0.00327

A (2, 2) = 0.659929 0.340071 0.148341 0.851659
B (2, 4) = 0.728936 0.16819 0.091188 0.011686 0.061483 0.314266 0.259672 0.36458

With more states the model tries to learn more than actually information given in the data (it overfits). Thats why it starts to perform more poorly by assigning states that shouldn't have meaning a probability at the expense of the correct sates. With less states the opposite would occur. The model would combine probabilities instead and might overestimate.
Which dimensionality combination is the best depends case by case (the complexity of the patterns to recognise) and amount of training data.

As a rule of thumb N-1 states with N = number of different observations might be a useful starting point for exploring the domain. But it mainly depends on the dimension of the underlying state space (degree of freedom of the problem domain)

QUESTION 10:
Uniform distributed initialisation. How does this affect the learning?
A(3, 3) = 0.333333 0.333333 0.333333 0.333333 0.333333 0.333333 0.333333 0.333333 0.333333
B(3, 4) = 0.2642 0.2699 0.2085 0.2574 0.2642 0.2699 0.2085 0.2574 0.2642 0.2699 0.2085 0.2574
Matrix A does not change, since the normalisation in the baum welch algorithm keeps it constant.
While B just changes a bit, which we have no good explanation for yet.
The training process is basically not existing.

Diagonal distributed initialisation. How does this affect the learning?
A(3, 3) = 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
B(3, 4) = 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.2642 0.2699 0.2085 0.2574
The model never changes state from its initialisation, that’s why A has a definite outcome state as defined in pi.

Initialisation similar to solution. How does this affect the learning?
3 3 0.687347 0.100075 0.212577 0.106908 0.705591 0.1875 0.177061 0.277032 0.545907
3 4 0.633074 0.186943 0.134895 0.045087 0.131602 0.494387 0.245997 0.128014 0.063408 0.065393 0.234409 0.63679

In case of a really small difference in values it quickly converges. (Finding of Carlos)