Q1:
Build Table A (NxN size):
For each state, obtain its transition frequency to all other states (i.e. n(i)). For e.g. For state0, freq(0->0) +freq(0->1) + freq(0->2)+freq(0->3) +freq(0->4)… For state1, freq(1->0)+ freq(1->1)+ freq(1->2)+…

Then calculate possibility of state transitions. For transition which end up with 'BEGIN' and 'END', set possibility to 0.0. For known transitions, calculate as given in specification. Unkown ones, as well.

Then obtain, Pi, the initial possibility distribution.

Build Table B ():
Like build table A, obtain total frequency of one state emits all symbols (i.e. n(i)). Set n('BEGIN') and n('END') to 0.

For each emission end up with 'UNK'. Apply plus one smooth. For else emission possibility, calculate as stated in specification.

Then for each query, apply Viterbi algorithm on it.
In Viterbi algorithm implementation, we used a list of dictionary named "Vit". At each time stamp, the largest possibility of this state is recorded, e.g. Vit[1][2] means the largest possibility of state2 transit from time stamp 1.

Calculate Vit[0][all_states] at the beginning, hence simply Pi[state_id] * B[state_id][symbol_id]. B[state_id]['UNK'] for 'UNK' symbols. Since we were calculating from BEGIN, so first element of state sequence is the state_id that we just used.

Then from time stamp 1 all the way to last time stamp. At each time stamp, for each state, calculate the probability of all states in last time stamp transits to current state in current time stamp. So we have Vit[t-1][y0] * A[y0][y] * B[y][symbol at current time stamp] in which y0 is state in last time stamp, y is state in current time stamp. This gives numbers of possibilities hence use max() to pick the optimal solution. When it comes to last time stamp, we need to multiply possibility of current state transits to 'END' state. After calculating optimal solution for each state at one time stamp, append current state to state sequence.

Upon finishing walking through all time stamps, append 'BEGIN' and 'END' to complete state sequence. Then apply log on the result possibility. Question1 complete.

Q2:
Original Viterbi Algorithm has found the most likely state sequence for given symbol sequences. In this question, we need extend it to have top-k likely state sequences. In most extremely cases, each sequence comes from the same tail state.

Therefore, we need to build list of size k to store top-k state sequence for time series T.
For the first few time series, we are facing a challenge that the number of top likely state sequences is less than k. In this case, we have to find some value to fill in the list so that probability less than or equal to 0 resulting in the list won't be neglected.

For each time, we generally need to pick up top-k state sequences out of k*number of states. Finally we may pick up top-k results at very end of query.

Q3:
In this section we need to change methodology of smoothing, to improve accuracy of prediction. Here we used good-Turing:
For emission matrix B:

$$C^* = (c+1) * \frac{N_{r+1}}{N_r} , \qquad \text{Prob} = \frac{C^*}{\sum C^*}$$

Where c is the frequency of (state, symbol). $N_r$is the number of (state, symbol), whose frequency == r.

For probability, dominator is sum of all (state, symbol), whose state == specific state y_0.
By good-Turing smoothing, our accuracy could reach 89%, which is probably a better solution than add 1 smoothing.