Ozan Cinci
2448223

HW2

Q1.

a)

(a (b+c)* a + aa + b) (a+b)*
first box -> (b+c)
second box -> aa
third box -> b
fourth box -> (a+b)*

b)

A = 0
B = 1
C = 0,1
D = 2
E = 1
F = 0,2

Q2.

a)

It is the algorithm that is used for “State Elimination”. It lets us lower the number of states while maintaining the language capability of DFA. The basic idea of the algorithm is to merge states in the DFA until no further merges are possible. For each non-start accepting state in turn, eliminate the state and update the transitions according to the procedure. After eliminating all the possible transitions, we get one transition that is basically a regex expression.

b)

The starting conditions must be the same. If the starting state has an incoming transition then create a new start state which does empty transition to the old transition. Also, if there are multiple final states or a final state has outgoing transitions then create a new final state that has empty incoming transitions from previous final state(s). We can evolve this algorithm such that:

- In the new machine, we have accepting and rejecting states. Further explanation is done on the next step
- Since there is no explicit accept or reject state, we assume all the possible states are final states because if a string stays one of the states after consuming, we can say that it is one of the accepted strings and then this state is the final state. Referring back to the “State Elimination Algorithm” we should create a new final state that has incoming empty transitions from all of the states (and has no outgoing transitions) so that if a string stays in one of the previous final states, then it is accepted by the new algorithm thanks to fact that they can do empty transition to new final state.
- Then current states have an empty transition to only and single one final state.
- Since we want to focus the output string the automata can create, we can ignore the inputs and treat possible output letters as input letters. For the sake of easy comprehension, we can only calculate and draw outputs on the figure.

- After the elimination procedure, we will have only two states that we created. The new start state and the new final state.
- The only and single transition we have is the regex of what we can get as an output.

(I try my best to explain the algorithm in my mind but if it is not clear, option C has step by step drawing)

c)

Before starting:

There is only one way to end with C and this is the transition that can be done from q2 to q1. So as long as we end up in the state q2, we can append C to the and. So there is a small modification for this problem. Instead of calculating all possible cases, we only calculate the case where we end up in q2 so we can append C to the end.

Step1)

All the previously given states are final states then I create a new start state and final state. The new final state is a state that accepts empty transitions from previous final states. Now we have only one final state.

Step2)

I remove all the empty transitions except one to simplify calculations.

Step3)

Eliminate q0.

Step4)

Eliminate q1.

Step5)

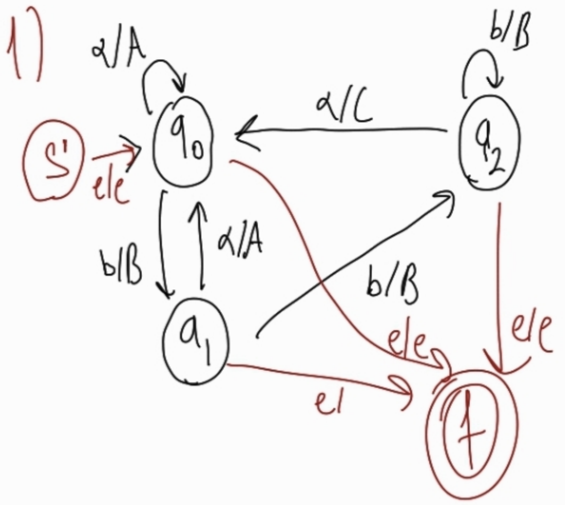
Eliminate q2. The only transition we have is the regex for all the possible outputs that ends with C.

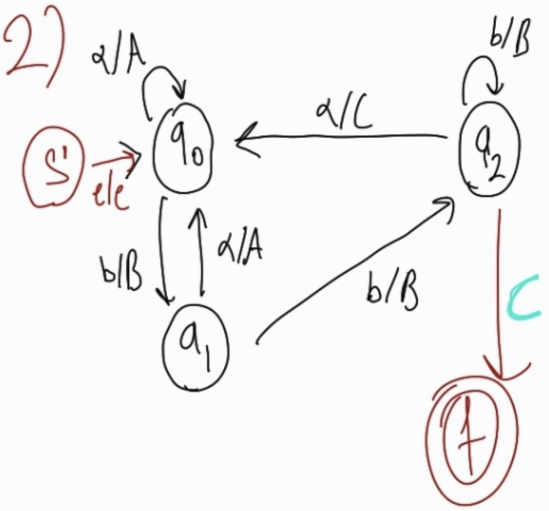
(please see the next page for drawing)

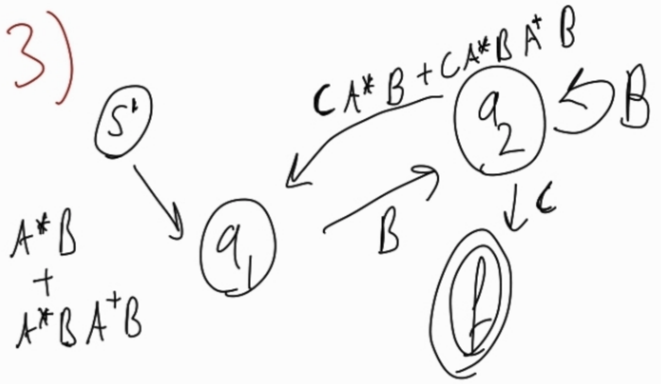
The answer is:

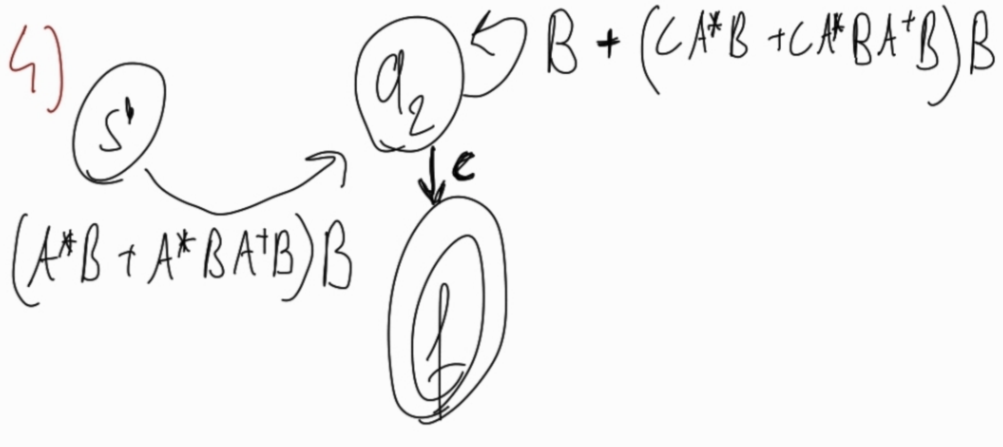
(A*B + A*BAA*B) B (B + (CA*B + CA*BAA*B)B)* C

(please see the next page for drawing)









$$\boxed{(A^*B+A^*BA^+B)B\left(B+(CA^*B+CA^*BA^+B)B\right)^*C}$$

Q3.

I could not solve Q3. But I found a way to merge N2 and N3. There will be a new start state that has empty transitions to N2 start and N3 start then I will convert NFA to DFA. But that is all I could think of.