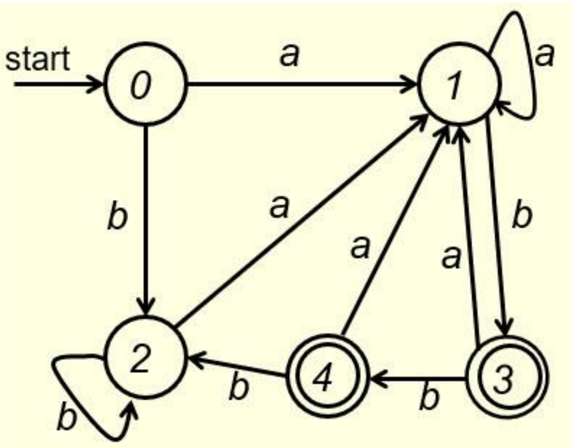
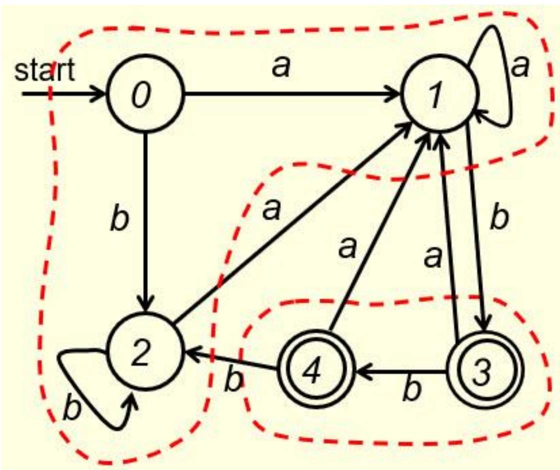
Solution Set of CS375 Homework Assignment 4-1 (22 points)

Due date: 02/11/2025

1. Given the following DFA, to find a minimum-state DFA, (6 points)



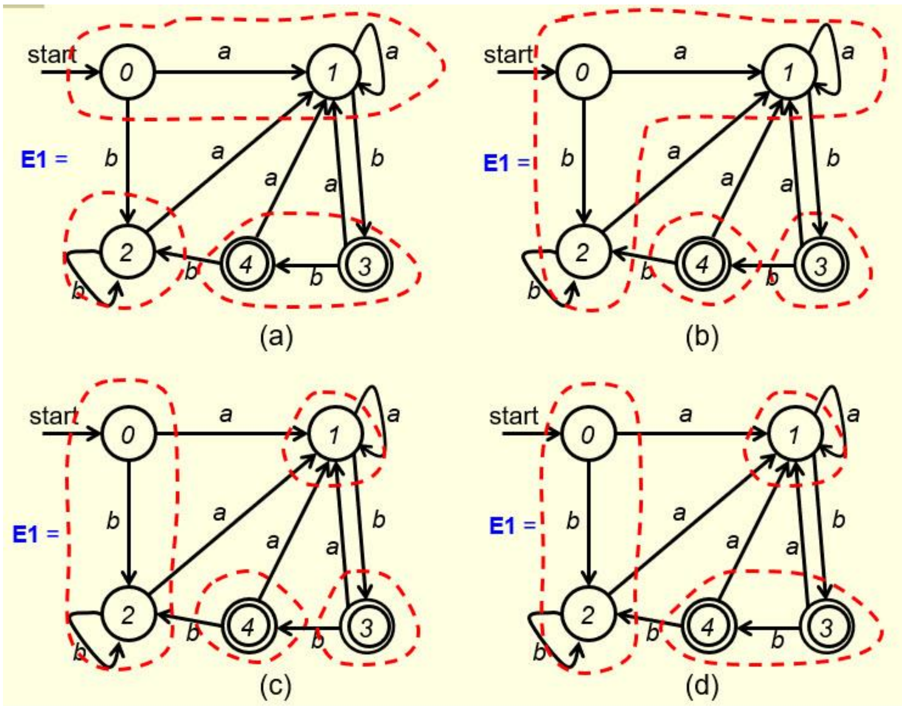
we first construct E_0= {{3,4}, {0,1,2}} as follows.



E_0 consists of two components, the set of non-final states {0,1,2}, and the set of final states {3,4}. We then construct E_1. Which one in the following four cases is the correct E_1? Put your answer in the following box.

(c)

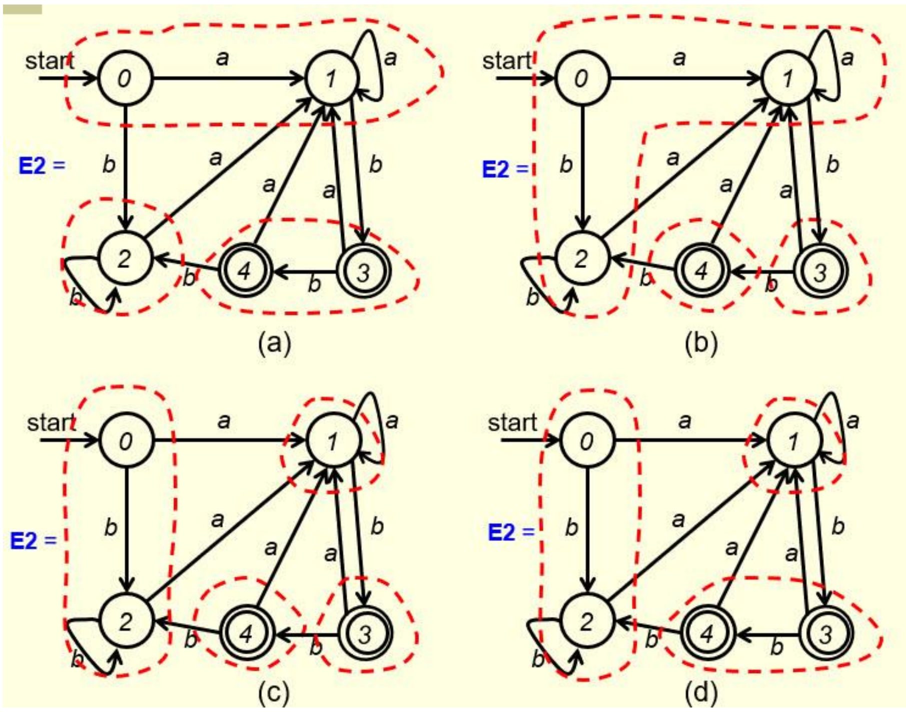
(1 point)



Next we construct E_2. Which one in the following four cases is the correct E_2? Put your answer in the following box.

(c)

(1 point)



Since $E_2 = E_1$, the states of the minimum-state DFA are:

[0], [1], [3], [4] or [2], [1], [3], [4]

(2 points)

or

{0, 2}, {1}, {3}, {4}

(You may use either partition component notation or equivalence class notation to represent the states)

The start state is: [0] / {0, 2} (0.5 points)

The final state(s) is/are: [3], [4] / {3}, {4} (1 point)

2. Let the set of states for a DFA be S = {0, 1, 2, 3, 4, 5}, where the start state is 0 and the final states are 2, 3 and 4. Let the equivalence relation on S for a minimum-state DFA be generated by the following set of equivalent pairs of states:

{ (2, 3), (0, 5) }

The states of the minimum-state DFA are: (2 points)

{0, 5}, {1}, {2,3}, {4}

(component notations)

or

[0], [1], [2], [4] or [5], [1], [3], [4]

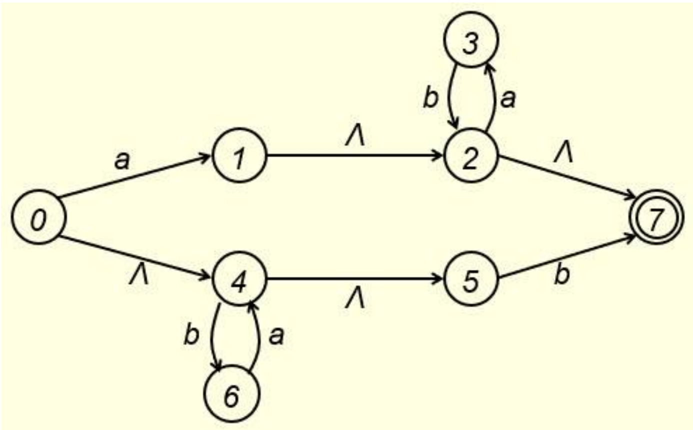
(Equivalence class notations)

You only need to answer one of the above two cases.

3. Given the following regular expression over the alphabet {a, b},

a(ab)* + (ba)*b

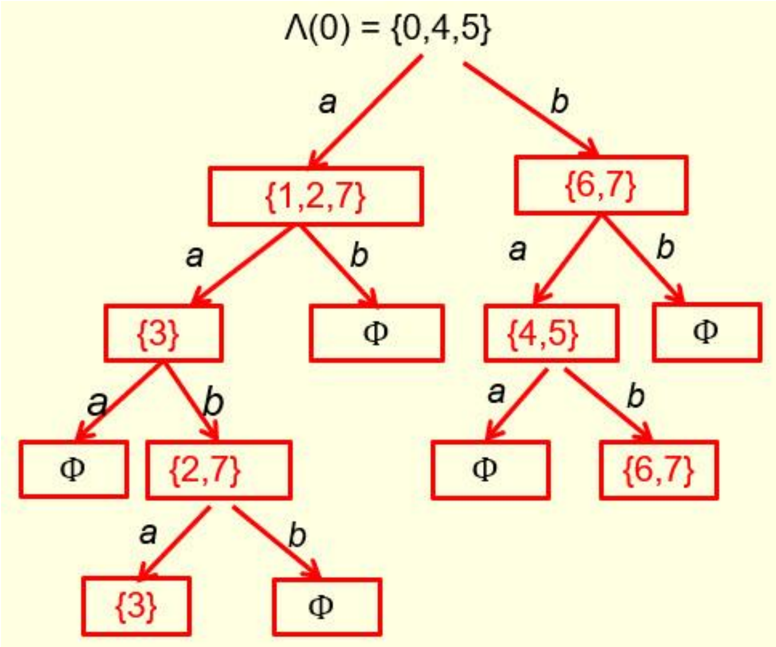
to transform it to a regular grammar, we first transform it to an NFA as the one shown below.



Instead of converting this NFA to a regular grammar directly, we convert it to a DFA first and then convert the DFA to a regular grammar (why?). So we construct Λ-closures of the above NFA,

Λ(0) = {0,4,5} Λ(1) = {1,2,7} Λ(2) = {2,7} Λ(3) = {3}
Λ(4) = {4,5} Λ(5) = {5} Λ(6) = {6} Λ(7) = {7} Λ(Φ) = Φ

and build the following tree, (3.5 points)



so the following nodes of the tree can be used to build a DFA. Fill out the following blanks and blanks in the above tree. (1.5 points)

{0,4,5}	{1,2,7}	{6,7}	{3}	{4,5}	{2,7}	Φ

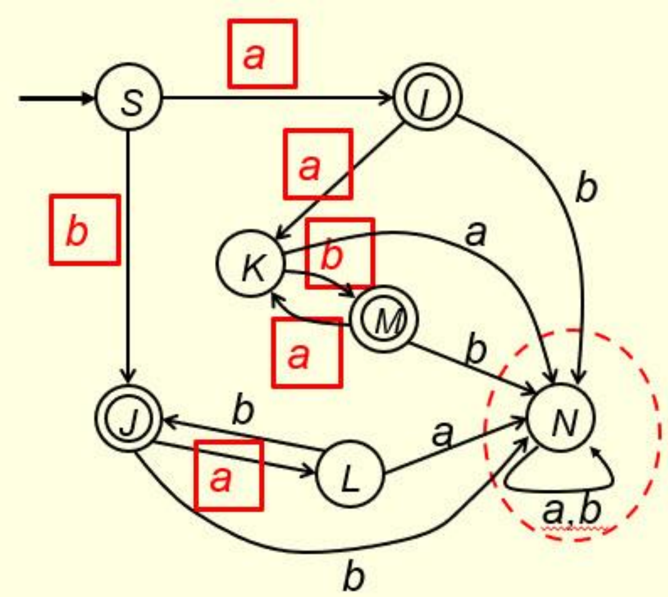
4. With the selected nodes from the above tree, we build a DFA by constructing the following transition table (the table on the left), and renaming the states as S (start state), I, J, K, L, M and N: (4 points)

	T	a	b
S	{0,4,5}	{1,2,7}	{6,7}
F	{1,2,7}	{3}	Φ
F	{6,7}	{4,5}	Φ
	{3}	Φ	{2,7}
	{4,5}	Φ	{6,7}
F	{2,7}	{3}	Φ
	Φ	Φ	Φ

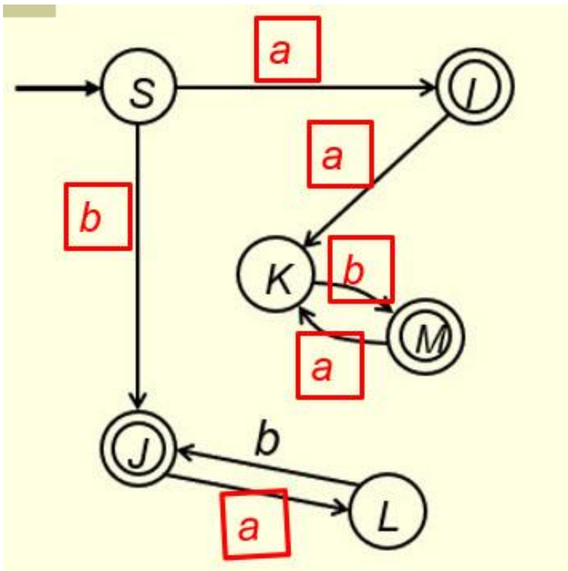
⇒

	T	a	b
S	S	I	J
F	I	K	N
F	J	L	N
	K	N	M
	L	N	J
F	M	K	N
	N	N	N

Then convert the table representation to a digraph representation as follows. Fill out the blanks in the following digraph. (1.5 points)



5. However, we don’t need the portion circled in the above digraph representation of the DFA (why?). By removing that portion, the DFA is like the one shown on the left side of the following figure. No points for this part.



$S \rightarrow$	aI	|	bJ
$I \rightarrow$	aK	|	Λ
$J \rightarrow$	aL	|	Λ
$K \rightarrow$	bM		
$L \rightarrow$	bJ		
$M \rightarrow$	aK	|	Λ

By constructing production rules from this FA, we get the production set of the regular grammar on the right side of the above figure. Fill out the blanks in the above figure. (3.5 points)