CS 340, Fall 2016 — Sep 29th — Exam 1 Name: Solution

Note: in all questions, the special symbol ϵ (epsilon) is used to indicate the empty string.

Question 1. [10 points] Specify a regular expression that generates the language over the alphabet {**a**, **b**} of all strings that end in either **aba** or **bab**.

Examples of strings in the language:	Examples of strings not in the language:
aba	ϵ
bab	**ab**
aaba	**ba**
abbabab	**babaa**
bbbbaba	**ababba**

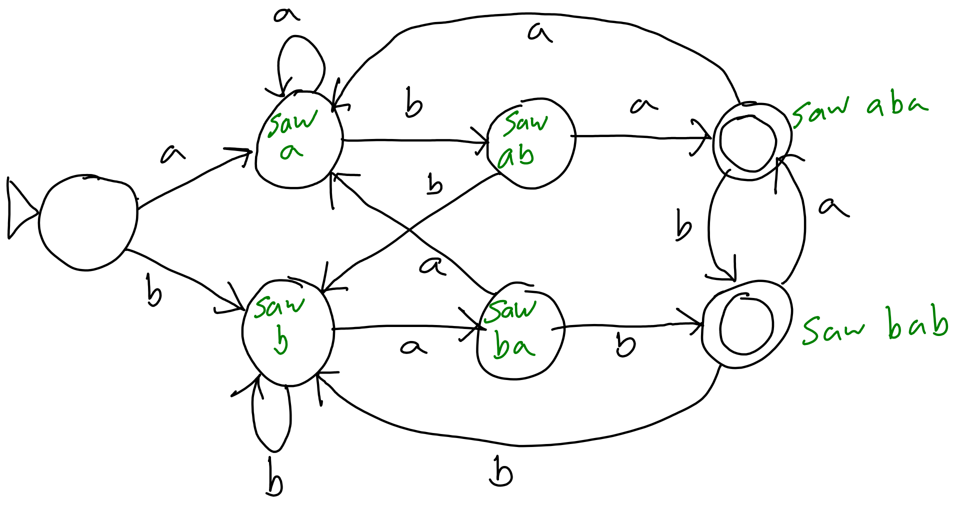
(a|b)*(aba|bab)

Question 2. [10 points] Specify a regular expression that generates the language over the alphabet {**a**, **b**} of all strings with an even number of **b**s.

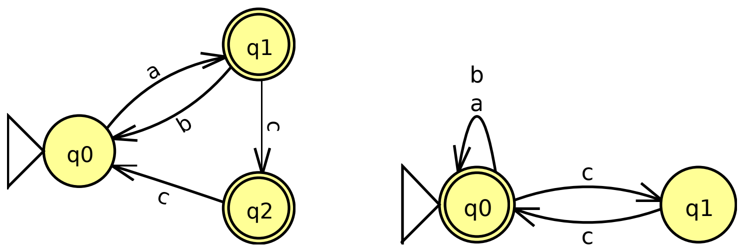
Examples of strings in the language:	Examples of strings not in the language:
ϵ	**b**
bb	**bbb**
abb	**aba**
abba	**abbaba**
abbabaab	**bbababab**

a*(a*ba*ba*)*a*

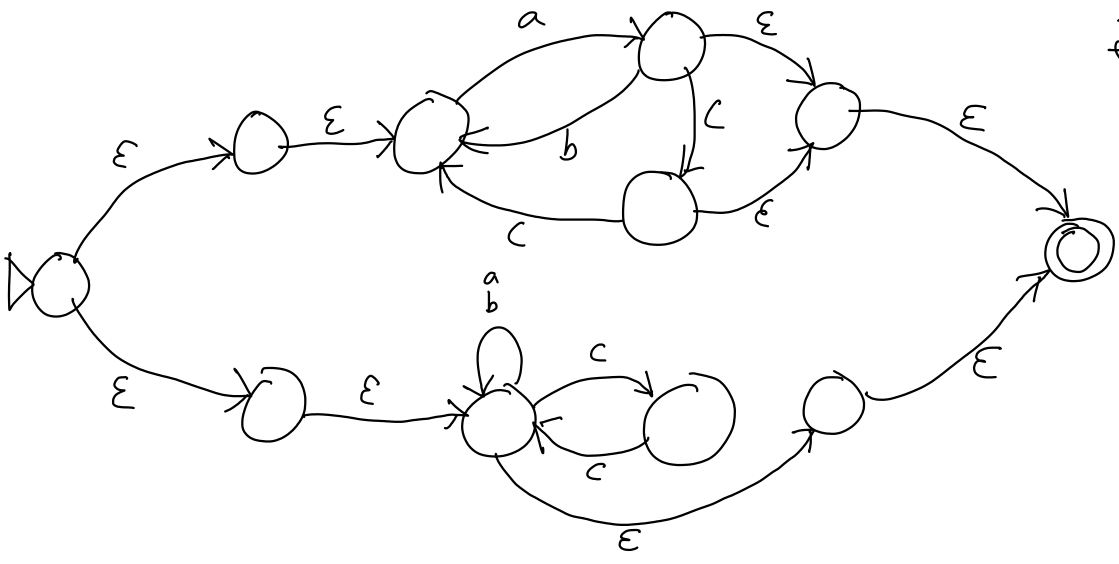
Question 3. [10 points] Create a *deterministic* finite automaton (DFA) that recognizes the language described in Question 1. (See that question for examples of strings in the language and not in the language.) Make sure you indicate the start state and the final state(s), and that each transition has a direction and is labeled with exactly one input symbol.



Question 4. [10 points] Consider the following two DFAs:

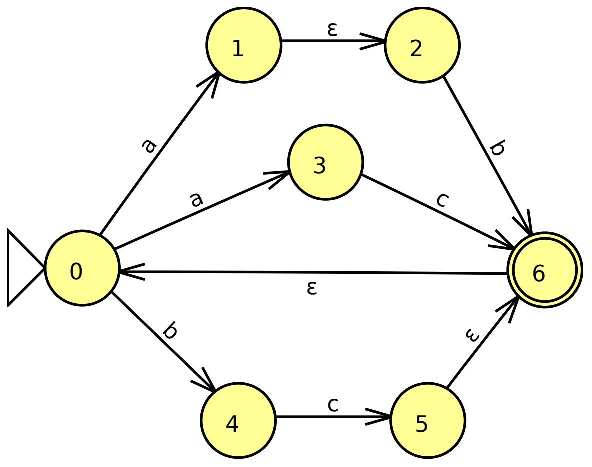


Show a finite automaton that recognizes the union of the languages recognized by these two automata. In other words, any string that is accepted by *either* of the automata should be accepted, and any string that is rejected by *both* automata should be rejected. **Hint**: your automaton does not need to be deterministic.



Idea is to execute both FAs in parallel: see Lecture 9

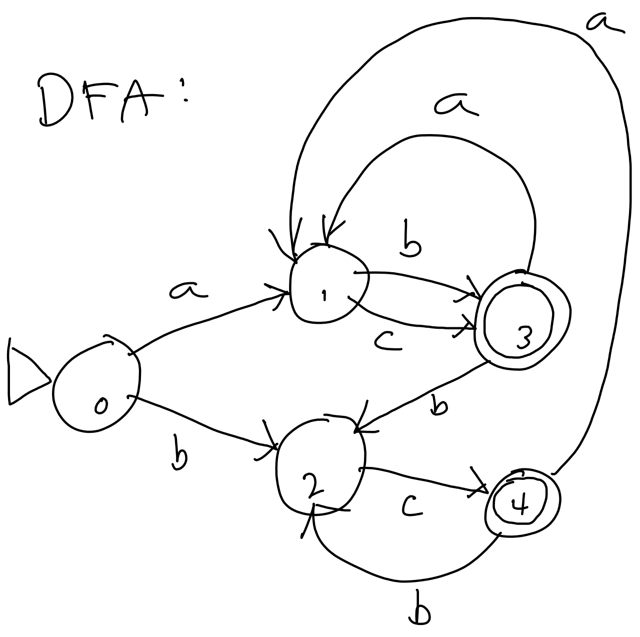
Question 5. [10 points] Consider the following nondeterministic finite automaton (NFA):


$$\begin{tikzcd}
 & 1 \arrow[r, "\varepsilon"] & 2 \arrow[rd, "b"] & \\
\arrow[r] & 0 \arrow[u, "a"] \arrow[r, "a"] \arrow[d, "b"] & 3 \arrow[r, "c"] & 6 \arrow[lll, bend left, "\varepsilon"] \\
 & 4 \arrow[r, "c"] & 5 \arrow[ru, "\varepsilon"] &
\end{tikzcd}$$

Convert this NFA into a deterministic finite automaton (DFA). Show the table mapping each reachable set of NFA states to an equivalent DFA state. Make sure your DFA indicates the start state, final state(s), and that each transition indicates a direction and is labeled with exactly one input symbol.

NFA states	DFA state
✓ {0}	0
✓ {1, 2, 3}	1
✓ {4}	2
✓ {0, 6}	3
✓ {0, 5, 6}	4

DFA:


$$\begin{tikzcd}
\arrow[r] & 0 \arrow[r, "a"] \arrow[d, "b"] & 1 \arrow[r, bend left, "b"] \arrow[r, "c"'] & \boxed{3} \arrow[l, bend right, "a"'] \arrow[ld, "b"] \\
 & & 2 \arrow[r, "c"] & \boxed{4} \arrow[l, bend left, "b"] \arrow[uul, bend right, "a"']
\end{tikzcd}$$

Question 6. [10 points] Create a context-free grammar (CFG) for the language of *ALists*. The terminal symbols are $\boxed{\textbf{( ) , a}}$. An *AList* is a pair of balanced parentheses containing a comma-separated list of one or more *AItems*. Each *AItem* is either the terminal symbol **a** or an *AList*.

Examples of strings in the language:	Examples of strings not in the language:
(a)	ϵ
(a , a)	**a**
(a , (a))	**( )**
((a))	**((a)**
((a , (a , a)) , (a))	**(a a)**

Make sure to indicate what the start symbol of your grammar is. L is the start symbol

$L \rightarrow (Z)$

$Z \rightarrow I$

$Z \rightarrow I , Z$

$I \rightarrow a$

$I \rightarrow L$

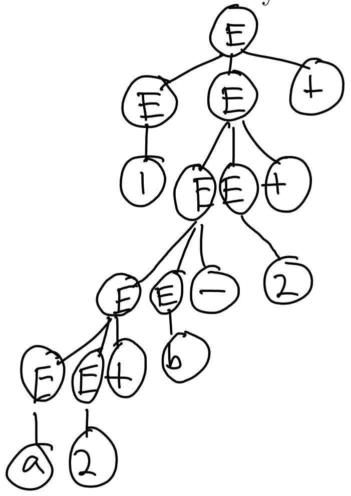
Question 7. [20 points] Consider the following context free grammar with the terminal symbols $\boxed{\textbf{a b 1 2 + -}}$ and the nonterminal symbol $\boxed{\text{E}}$, which is the start symbol:

E → **a**
E → **b**
E → **1**
E → **2**
E → E E **+**
E → E E **-**

(a) Show a derivation of the input string $\boxed{\textbf{1 a 2 + b - 2 + +}}$. (Continue on the right-hand side if necessary.)

String	Production
E	E → EE+
EE+	E → 1
1 E+	E → EE+
1 EE++	E → EE −
1 EE − E ++	E → EE+
1 EE+E −E++	E → a
1 aE+E − E++	E → 2
1a 2+E−E++	E → b
1a2+b−E++	E → 2
1a2+b−2++	

(b) Draw the parse tree for the derivation you found in part (a).



Question 8. [10 points] Consider the following productions on the nonterminal symbol G:

$G \rightarrow a\ P$
$G \rightarrow b\ Q$
$G \rightarrow c$

Assume that **a**, **b**, and **c** are terminal symbols and G, P, and Q are nonterminal symbols.

Show the pseudo-code for a recursive descent parse function for the nonterminal G (*i.e.*, a `parseG` function).

Important: Show how to use the lexical analyzer to choose between the possible productions. Also, show how to raise an error if no production can be applied.

```
parseG() {
        t = lexer.peek()
        if (t == 'a') {
                lexer.next()
                parseP()
        } else if (t == 'b') {
                lexer.next()
                parseQ()
        } else if (t == 'c') {
                lexer.next()
        } else {
                throw new ParserException
        }
}
```

Question 9. [10 points] Design a Turing Machine that takes a string of **a** and **b** symbols as input, and accepts the input (by halting) if and only if there are more occurrences of **a** than occurrences of **b**.

For example, the strings **aba** and **aababba** would be accepted, but the strings **abb** and **aababb** would be rejected.

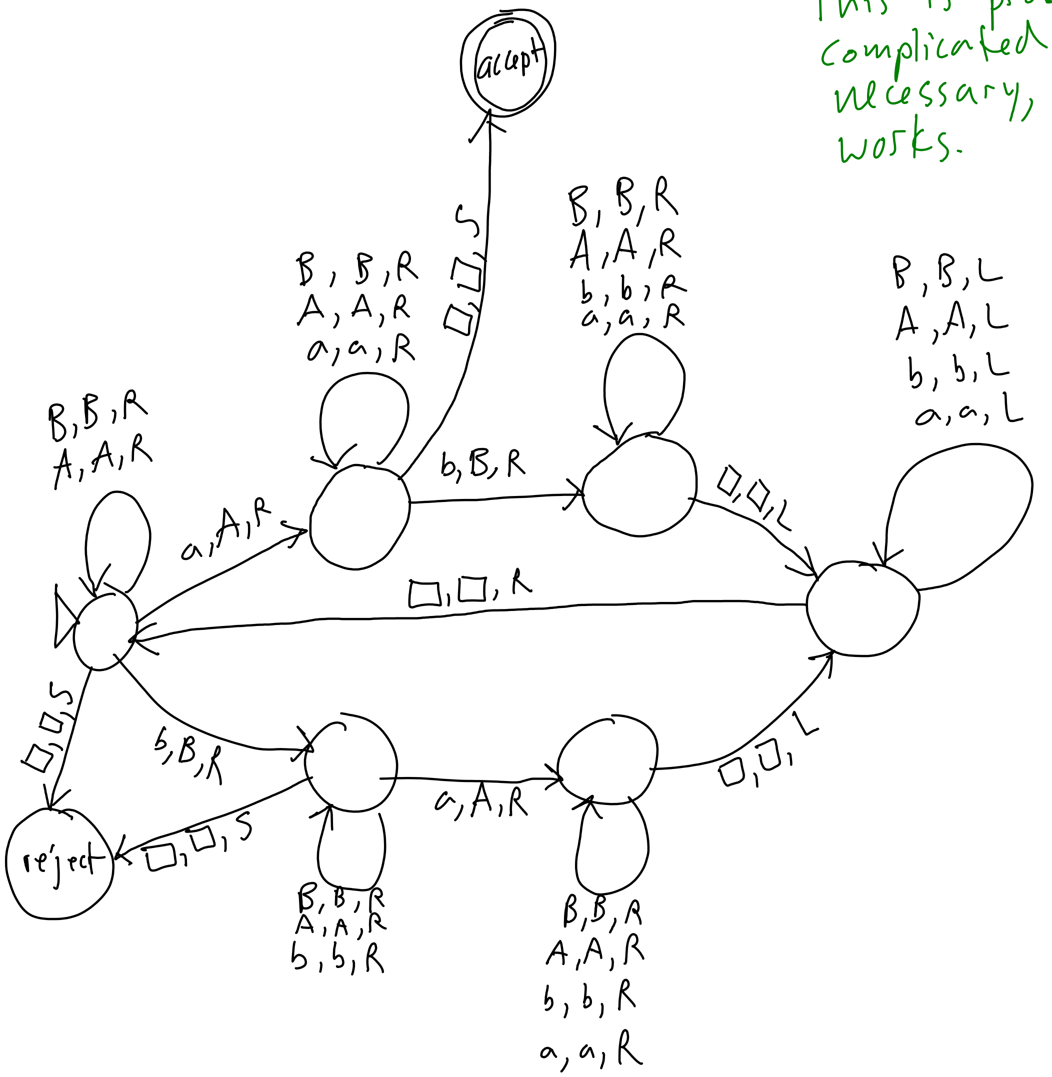
(a) Start by stating in words how you could make such a Turing Machine work.

Match pairs of a/b, change them to A/B

If we run out of bs before we run out of as, accept/halt.

(b) Show the states and transitions of the actual Turing Machine. (Do as much as you can; substantial partial credit will be awarded if you are on the right track.)

This is probably more complicated than necessary, but it works.


accept
□,□,S
B,B,R
A,A,R
a,a,R
B,B,R
A,A,R
b,b,R
a,a,R
B,B,L
A,A,L
b,b,L
a,a,L
B,B,R
A,A,R
a,A,R
b,B,R
□,□,L
□,□,R
□,□,S
b,B,R
a,A,R
□,□,L
reject
□,□,S
B,B,R
A,A,R
b,b,R
B,B,R
A,A,R
b,b,R
a,a,R