CS352 - Compilers: Principles and Practice
Homework 1

Due: 2004 October 19, beginning of lecture

1. Draw the finite state machines, and write regular expressions that denote the strings recognized by the following finite automata:

   (a) s_1 is the start state; s_2 is the final (accepting) state

	a	b
s_1		s_2
s_2	s_3	
s_3		s_2

   Answer: $b(ab)^*$

   (b) s_1 is the start state; s_4 is the final state

	a	b
s_1	s_2	s_3
s_2		s_4
s_3	s_4	
s_4		s_4

   Answer: $(ab|ba)b^*$

   (c) s_1 is both the start state and the final state

	a	b	c
s_1	s_2	s_3	s_4
s_2	s_1		
s_3		s_1	
s_4			s_1

   Answer: $(aa|bb|cc)^*$

   (d) s_1 is the start state; s_1 and s_3 are final states

	a	b	c
s_1	s_2		
s_2		s_2	s_3
s_3			

   Answer: $\epsilon|ab^*c$

2. Write DFA's that recognize the following languages. Be careful to note starting and accepting states in your automata.

   (a) $\{w \in \{a, b\}^* : \text{each } a \text{ in } w \text{ is followed by at least one } b\}$

   Answer:

	a	b
s_1	s_2	s_1
s_2		s_1

   s_1 is start and accept state.

   (b) $\{w \in \{a, b\}^* : w \text{ contains precisely three } a\text{'s}\}$

   Answer:

	a	b
s_1	s_2	s_1
s_2	s_3	s_2
s_3	s_4	s_3
s_4		s_4

   s_1 is start state, and s_4 is accept state.

   (c) $\{w \in \{a, b\}^* : w \text{ has an odd number of } a\text{'s and an even number of } b\text{'s}\}$

   Answer:

	a	b
s_1	s_2	s_4
s_2	s_1	s_3
s_3	s_4	s_2
s_4	s_3	s_1

   s_1 is start state, and s_2 is accept state.

   (d) $\{w \in \{a, b\}^* : w \text{ has } bab \text{ as a substring}\}$

   Answer:

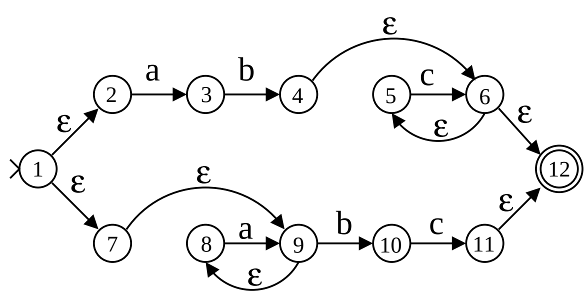
	a	b
s_1	s_1	s_2
s_2	s_3	s_1
s_3	s_1	s_4
s_4	s_4	s_4

   s_1 is start state, and s_4 is accept state.

3. Consider the following regular expression:

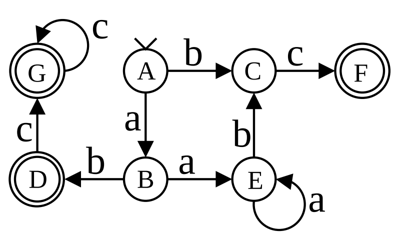
$$(abc^* \mid a^*bc)$$

(a) As described in class, construct an NFA for this expression.

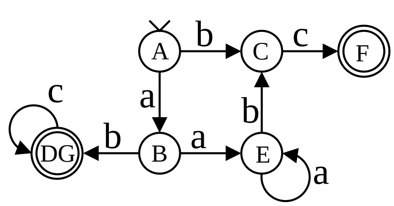


(b) Using subset construction, convert the NFA into a DFA.

	a	b	c
A {1,2,7,8,9}	B	C	
B {3,8,9}	E	D	
C {10}			F
D {4,5,6,12}			G
E {8,9}	E	C	
F {11,12}			
G {5,6,12}			G



(c) Optimize the DFA to minimize the number of states.



4. Consider the grammar below:

S	::=	S ; S
	|	`<ID> =` E
	|	`print (` L `)`
E	::=	E `+` E
	|	`(` S `,` E `)`
	|	`<ID>`
	|	`<NUM>`
L	::=	E
	|	L , E

(a) Write a grammar that accepts the same language as the grammar above, but that is suitable for LL(1) parsing.

1	S	::=	S' S''
2	S'	::=	`<ID> =` E
3		|	`print (` L `)`
4	S''	::=	; S
5		|	ϵ
6	E	::=	E' E''
7	E'	::=	`<ID>`
8		|	`<NUM>`
9		|	`(` S `,` E `)`
10	E''	::=	`+` E
11		|	ϵ
12	L	::=	L' L''
13	L'	::=	E
14	L''	::=	, E L''
15		|	ϵ

(b) Find *FIRST* and *FOLLOW* sets for your grammar.

γ	$FIRST(\gamma)$	$FOLLOW(\gamma)$
S	`<ID>` `print`	\$,
S'	`<ID>` `print`	\$, ;
S''	; ϵ	\$,
E	`<ID>` `<NUM>` `(`	\$, ;)
E'	`<ID>` `<NUM>` `(`	\$, ;) +
E''	`+` ϵ	\$, ;)
L	`<ID>` `<NUM>` `(`	)
L'	`<ID>` `<NUM>` `(`	) ,
L''	, ϵ	)

(c) Show the LL(1) parsing table for your grammar.

	`<ID>`	`<NUM>`	`print`	`+`	;	,	`(`	`)`	\$
S	1		1						
S'	2		3						
S''					4	5			5
E	6	6					6		
E'	7	8					9		
E''				10	11	11		11	11
L	12	12					12		
L'	13	13					13		
L''						14		15	

(d) Given the input string, "`x = 5; print(x, 6)`", use your parse table to expand the goal into a parse for the expression. Note which grammar production is applied at each step.