Complier Design, Handout No.1, Assignment 1

ALPHABETS and LANGUAGES

To understand the purpose of this course, let's consider the following example. English ***alphabets*** are a..z, A..Z. English **words** such as Hello, Bye, …. are formed by using English alphabets. Not all collection of alphabets forms a valid English word. For example, words mochas, gracias, … are words but are not English words. In short, group of letters make up words and group of words using English **grammar** make up a sentence. Certain groups of correct sentences make up a paragraph. What is important to note that your English professors determine which sentences are valid and which are not.

This situation also exists with computer languages. Consider language C++. The language alphabets are a..z,A..Z, 0-9, and _. Words in C++ are called reserved words such as *if*, *else*, *while*, …. Certain set of words and some special operators are recognizable statements such as: *while (a < b)* ……. Set of statements become a C++ program. The compiler (like your English professor) will check the grammar of statements, if all statements satisfied the C++ grammar, the program will be translated into machine language, if not the compiler issues error messages.

Definition. An ***Alphabet*** is a finite set of symbols denoted by Greek letter Σ (Sigma).
Example

a. English alphabets: Σ ={a,b,c,…..,z,A,B,C,….,Z}
b. Binary alphabets: Σ ={0,1}

Definition. Given Σ, a ***word*** over Σ is any string of symbols used from Σ to create the word.
Example

a. If Σ={a,b}, then w1=abb is a word over Σ. w2 =baba is also a word over Σ
b. Σ={ a, ba }, then w1=ba is a word over Σ, but word ab is not a word over Σ

Definition. Give Σ and word w over Σ. the ***length*** of w or |w| is the number of symbols in Σ used to create w.
Example

a. Σ ={a,b}, then the length of word w=aba or |aba|= 3 (the 3 symbols are: a, b, and a)
b. Σ ={a, ba}, then for word w=aba, |aba|= 2 (the 2 symbols are: a and ba. In Σ, ba is a symbol)

Definition. A word of length zero or ***null word*** is denoted by Greek letter λ (lambda). Therefore |λ|=0. If λ is part of a word, you can ignore it. For example, w=aλbλb= abb.

Definition. Given word w, then $w^0=\lambda$, $w^1=w$, $w^2=ww$, $w^n=www…w$. Hence, w^n means ***concatenation of*** w to itself n times.
Example.

a. Σ={ a, b}. For w=ab, $w^3=(ab)^3$ = ababab. Notice that $(ab)^3 \neq a^3b^3$=aaabbb
b. Σ ={ a, b, c}. Then $(cab)^2$ = cabcab $\neq c^2a^2b^2$ because $c^2a^2b^2$=ccaabb

Concatenation and Union of words: Given w1=ab and w2=a. Then w1w2=aba (***concatenation of w1 and w2***) is ONE word, but w1+w2 =ab +a, reads: ab or a is the ***union of w1 and w2 (union of two words)***. Hence w1+w1 = ab + ab={ab} U {ab} = ab (recall: in C++, the operator + between two strings act as a concatenation of two strings) , here + acts as the union of two strings. There are no duplicates in a set (all elements in a set are unique).

Words Factoring:

i. Let w1+w2 = ab + ac, since word ab begins with a, and also word ac begins with a, therefore both words have a in common on the left-hand-side, factor them by a: ab+ ac = a (b + c). To check your work, a(b+c) = ab + ac. This is called **left-factoring**.

ii. Let w1=ab and w2=bb. Both words end up at b, so we can factor b on the right-hand-side to get ab+bb = (a+b)b. This is called **right-factoring**.

iii. Let w1=abc and w2=adc. Both w1 and w2 begins with a, so we factor a on the left-hand-side to get abc+adc =a(bc + dc). The words in parenthesis have c in common on their right-hand-side, factor by c to get abc + adc = a (bc + dc)= a(b+d)c

iv. Let w1=ab and w2=a. By left-factoring we factor by a on the left-hand-side to get a(b + λ). To check the correctness of your answer, remove the parentheses by concatenation: a(b+λ) =ab + aλ= ab + a = w1 (note. aλ = a)

Definition. Given Σ. The set of all words over Σ including λ is denoted by **Σ*** (reads sigma star).
Example
Given Σ = { a, b }, to not miss any word, we list all words by their length

length	Length 0	Length 1	Length 2	Length 3	………..
words	λ	a, b	aa, ab, ba, bb	aaa, aab, aba, abb, baa, bab, bba, bbb	……..

Therefore, the set of all words using a and b including λ is= Σ* = { λ, a, b, aa, ab, ba, bb, …….. }

Definition. Given Σ, any subset of Σ* is called a ***Language***
Example
Let Σ={a, b}, then Σ*={ λ, a,b,a^2,ab,ba,b^2, …….. }.

i. Language L1 = {all powers of a including the 0 power }= { λ, a, aa, aaa, aaaa, ….}= {a^0,a^1,a^2,a^3,…} , we name this language a* or language whose words are all powers of a including the zero power, so {a^0, a^1, a^2, a^3,…} = a*
ii. Language L2 ={ all powers of a excluding the 0 power }={ a, aa, aaa, aaaa, ….}= { a^1,a^2,a^3,…} We can factor by a on the left-hand-side to get: =a{ λ, a^1,a^2,a^3,…} =aa*, or We can factor by a on the right-hand-side to get : { λ, a^1,a^2,a^3,…} a =a*a Therefore aa*=a*a = a^+ ,reads: a plus. Thus { a^1,a^2,a^3,…} = aa* =a*a = a^+
iii. Find the general from of L3 = { λ, ab, abab, ababab, ……. } = { $(ab)^0$, $(ab)^1$, $(ab)^2$, …. } = (ab)*
iv. Simplify L4 = a*b + a*a = a*(b + a) v. Simplify L5 = a + a* = {a } U {λ, a , a^2, a^3,… } = {λ, a , a^2, a^3,… } = a*

Examples. True or false?

i)a^3 ϵ a*, true reason: a*= { λ,a,a^2,a^3,.....}	ii)a^2b ϵ a*b*, true reason: a*b* = {λ,a,a^2,..}{λ,b,b^2,b^3,..} = {λ,a,b,ab,a^2b,... }	iii)a^2b ϵ a*+ b*, false reason: a* + b* ={λ,a,a2,... }U{λ,b,b2,... } ={λ,a,b,a^2,b^2,a^3,b^3........} a^2b is not in a*+b*
iv)a*b* = a*+ b*, false reason: ab ϵ a*b* but not in a*+ b*	iv)b^3 ϵ a*b*, true reason: a*b*, let a*=λ = λb* = b*={λ,b,b^2,b^3,..}	vi)$a^2b^3a^2$ ϵ a*b*a*, true reason: from a* choose a^2, from b* choose b^3, and from the last a* choose a2. Hence $a^2b^3a^2$ ϵ a*b*a* vii)Is b^3 ϵ a*b*a* ?, true Reason: λb^3λ = b^3

Example. Expand (a+b)* to show the words of the language

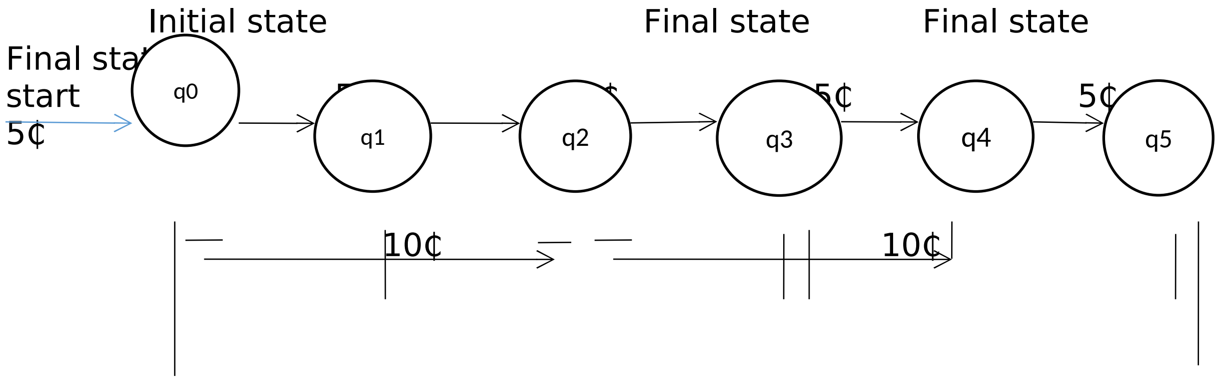
(a+b)* = { $(a+b)^0$, $(a+b)^1$, $(a+b)^2$, $(a+b)^3$, ... }
={ λ , a,b, (a+b)(a+b), (a+b)(a+b)(a+b)...............}
={ λ , a, b, aa,ab,ba,bb, (aa, ab, ba, bb)(a+b), ...}
={ λ , a,b, a^2,ab,ba,b^2,a^3,aab,aba,abb,baa,bab,bba,b^3,.... ....}
={λ,a,a^2,a^3,.......,b,b^2,b^3,...,ab,ba,................ }
= set of λ , all powers of a, all powers of b, and any combinations of a's and b's

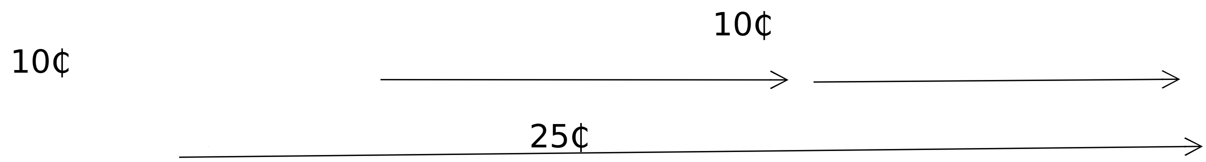
All of the following are true:

a^3 ϵ (a+b)*, b^5 ϵ (a+b)*, a^3b^2 ϵ (a+b)*,$a^2b^3a^3$ ϵ (a+b)*, abb ϵ (a+b)*, λ ϵ (a+b)*,

FINITE AUTOMATA (FA)

Suppose we want to design a vending machine to accept 5, 10, and 25 cents (¢) coins and return items which worth a total of 10, 15, or 25 cents. To make it a simple machine, assume the machine does not return any change and you can get something back from the machine if the exact chain of coins you drop in the machine is exactly the price of the item you want to receive.





This is an example of a Finite Automata (or machine with finite number of states). Finite Automata (FA) consist of the following components :

(i) **Set of states** ={ q0, q1, q2,q3, q4, q5 }, with only ***ONE initial state {q0}***, one or more ***final states {q2,q3, q5 }*** or states you can get items back from the machine, and zero or more ***NULL states {q1,q4 }*** or states that are not producing any output (not final nor initial state).

(ii) **Machine Alphabets=** Set of inputs= Σ={5¢, 10¢, 25¢ }, these are the only type of coins we can drop in the machine to go from one state to another state.

(iii) **Machine language: L**= set of chain of coins you can insert in the machine to enter a final state (get an item back from the machine) = { for simplicity list the chain of coins based on their length}

No. of coins	1	2	3	……..	5
String of coins (words)	(25)	(5)(5), (10)(5), (5)(10)	(5)(5)(5),(5)(10)(10), (10)(5)(10),)10)(10)(5)		(5)(5)(5)(5)(5)

Machine Language: L={ (25), (5)(5), (10)(5), (5)(10),……., (5)(5)(5)(5)(5)}

NOTE, pay attention to (10)(5) and (5)(10) chains. They are two different forms of dropping coins in the machine and that's why we have to consider them as two different chains of input.

(iv) **Set of rules: Instead of drawing FA, t**here are two other methods to provide rules on how to go from one state to another state: ***Transition table*** and ***Machine Grammar***

Transition table

states	**q0**	**q1**	**q2**	**q3**	**q4**	**q5**
q0		5	10			25
q1			5	10		
q2				5	10	
q3					5	10
q4						5
q5						

The highlighted 5, means from q0 coin 5 cents take us to state q1. The highlighted 10, means from q1 coin 10¢ will take us to state q3

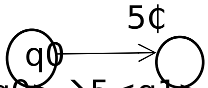
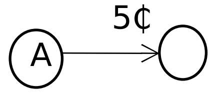
Machine Grammar

<q0>→ 5<q1>
Means from state q0, if we drop 5¢ we will enter state q1

<q1>→10<q3>
Means at state q1, if we drop 10¢ we will enter state q3

Notations:

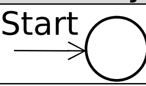
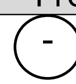
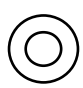
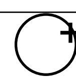
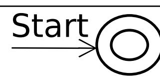
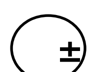
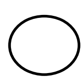
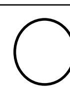
a. If you use a lower case letters to name a state, then in grammar rules we enclosed the name of the state within < and >. If you use upper case letters the < and > are not required anymore.
Example

using lowercase letters to name states:q0,q1	using uppercase letters to name states: A, B
5¢  q0 q1 , <q0>→5<q1> must use < and > in the machine grammar	5¢  A B , A→5B Don't use the angle brackets for A and B

b. To identify a final state in grammar, we use this notation

More grammars of the above FA	
<q0>→5<q1> <q0>→10<<q2> <q1>→5<q2> <q1>→10<q3> <q2>→5<q3> <q2>→10<q4> <q2>→λ, q2 is a final state	<q3>→5<q4> <q3>→10<q5> <q3>→ λ, q3 is final state <q4>→5<q5> <q5>→ ,q5 are final states All final states must have a λ on the right hand side of the grammar

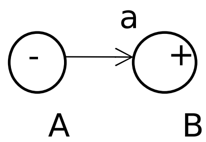
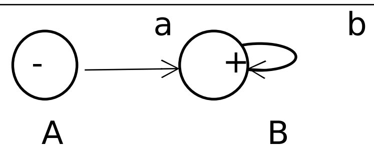
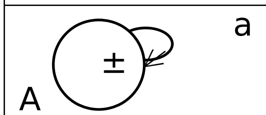
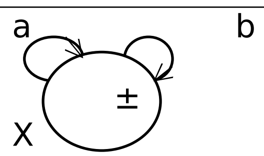
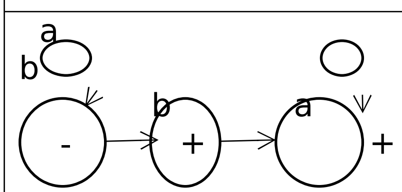
c. Instead of labeling "Initial state" and "Final state", we will use the following notations:

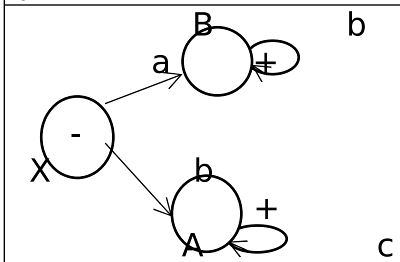
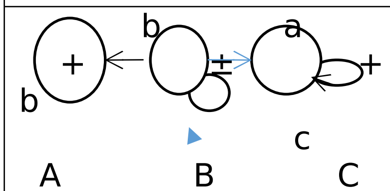
Name	symbols	Preferred symbols
Initial state	Start 	 - , minus sign
Final State		 + **,** plus sign
Initial and final state	Start 	 ± **,**plus minus sign
Null state		

FINITE AUTOMATA, LANGUAGES AND GRAMMARS

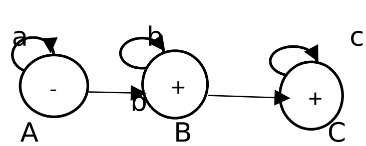
Examples to understand the concepts of FA's , Grammars and Languages

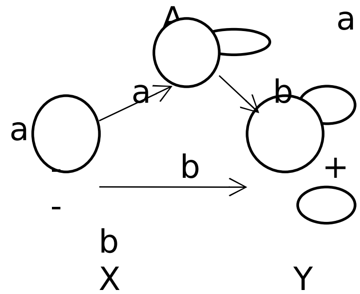
Example. Find the language and grammar of the following FA's

Give: FA	Find: Language	Find: Grammar
	∑={ a}, only input "a" take us from Initial state to final state Language:: L={a}	A→aB, at state A input a takes us to state B. The state on the left of the 1st Grammar rule is the *<u>initial</u>* *<u>state</u>* B→λ , B is a finial state
	∑={a, b} L={a, ab, abb, abbb, ... } ={ a, ab, ab^2, ab^3, ...} =a{λ,b, b^2, b^3, ...}=ab*	A→aB, A is the initial state B→bB, at B input b goes back to B B→λ , B is final state
	∑={a} L={**λ** ,a, a^2, a^3 , ...}=a* Note. When initial state is also final state, then λ is a word in the language of the machine.	A→aA, a loop means you can skip it (to get λ), or go through it as many times as you want A→λ
	∑={a,b} L={λ, a, b, aa,ab,ba,bb,…} ={$(a+b)^0$, $(a+b)^1$, $(a+b)^2$,.} =(a+b)* Means all combinations of a's and b's in any order	X→aX, X is the initial state X→bX, loop at X X→λ, X is a final state
	∑={a,b} FA has 2 final states, means the language has 2 parts. One ends at B (L1=a*b) and the other	A→aA A→aA |bB A→bB B→λ B→aC | λ B→aC C→bC C→bC | λ

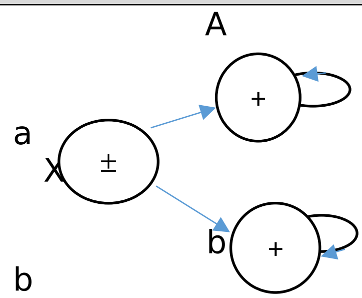
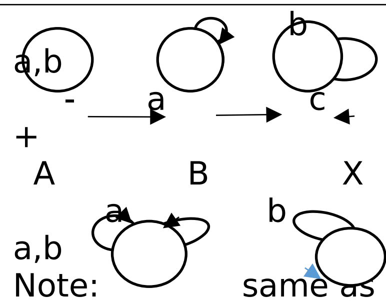
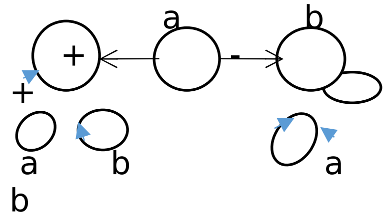
A B C There are two final states, means the final languge consist of two parts	ends at C (L2=a*bab*). So all together L=L1 + L2 =a*b + a*bab* = a*b(λ +ab*)	C→λ If two or more grammars have the same state on their left side, you can write them all on one line by using "|"
	∑={a,b,c} B is final: L1= ab* A is final: L2=bc* Hence L= L1 + L2 =ab* + bc*	X→aB X→bA X→aB | bA B→bB B→λ B→bB | λ A→cA A→λ A→cA | λ
	Σ={b, a, c} B is initial and final: L1=c* Which also includes λ A is final: L2= c*b, you can have zero or more c's before going to A C is final: L3==c*ab*. Thus, the language is: L=L1 +L2+L3= c* +c*b + c*ab* = c*(λ+ b + ab*)	B→cB | aC | bA | λ From B we can go to C,A, or just stop at B C→bC |λ From C we can go back to C or just stop at C A→λ

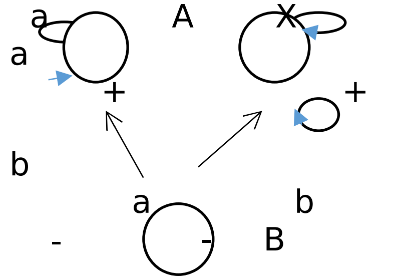
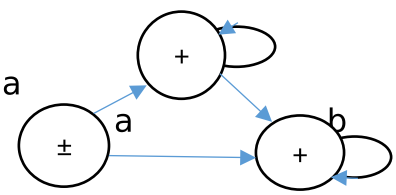
So far, we did some examples to find the language and the grammar when FA is given. Now let's look at some examples in which the grammar is given and we want to find the FA and the language of that grammar.

Given: Grammar	Find: FA	Find: Language
A→aA, A→bB B→bB, B→aC, B→λ C→cC, C→λ 3 states: A,B,C		∑={a,b,c} 2 final states: L1=a*bb*, L2=a*bb*ac* L=L1+L2 =a*b + a*bb*ac* =a*b(λ +b*ac*)
X→bB, X→λ B→bB, B→λ	b	∑={a.b} L1=λ ,X is initial and

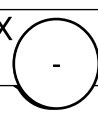
2 states: X and B	± b + X B	final L2=bb* ,B is final L=L1+L2 =λ + bb* = b*
X→aA, X→bY A→aA, A→bY Y→aY, Y→bY, Y→λ 3 states: X, A, Y		∑={a,b } 1 final but 2 ways to get to it L1=b(a+b)* L2=aa*b (a+b)* L=L1+L2 =aa*b(a+b)* + b(a+b)* =(aa* + λ)b(a+b)*=a*b(a+b)*

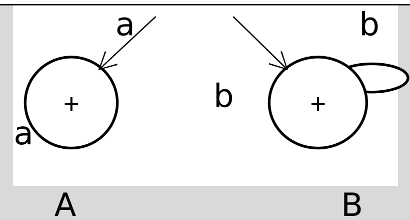
Now, suppose the Language is given and we want to find the FA, and then use the FA to find the Grammar of the language.

Given: Language	Find : FA	Find: Grammar
L = a* + b* Σ={ a, b } Two set of words, therefore machine must have 2 final states A and B, Since λ is in the language, initial state is also a final state		X→aA | bB | λ A→aA | λ B→bB | λ
L=ab*c(a+b)* ∑={a,b,c} Only one set of words, hence Only ONE final		A→aB B→bB, B→cX X→aX, X→bX, X→λ
L= a(a+b)* + b(a+b)* ∑={a,b} 2 set of words, hence 2 final states		A→aX, A→bY X→aX, X→bX, X→λ Y→aY , Y→bY, Y→λ

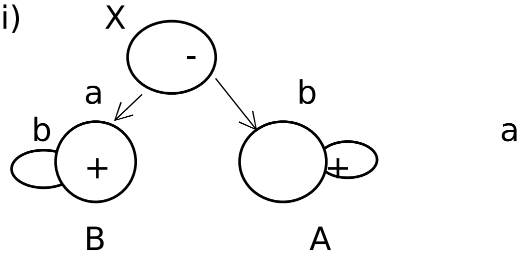
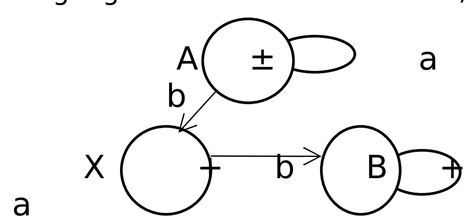
	X A Y	
L= aa* + b(a+b)* ∑={a,b} 2 set of words requires 2 final states		B→aA, B→bX A→aA, A→λ X→aX, X→bX, X→λ
L = a*b* Σ= { a, b } Even though there is only one set of words, but the machine must have 3 finals. One to get λ, another one all a’s and the last one to get a’s and then b’s, or just b’s Finals: X, A and B Initial state X which is also final state because λ is in L		X→aA | bB | λ A→aA | bB |λ B→ bB | λ Note. If we write the language of this machine, we get L1=λ, X is final L2=aa*, A is final L3=bb* + aa*bb*, B is final L=L1 + L2 + L3 =λ+aa* + bb* + aa*b* = aa* +aa*b* + λ+ bb* = aa*(λ+b*) + (λ+bb*) =aa*b* +b* = (aa* + λ) b* =a*b*

Example. Now, let’s put all these cases together and complete the following table. The shaded boxes are given, complete the table by filling out all empty boxes

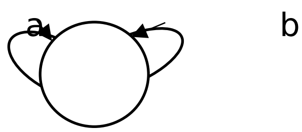
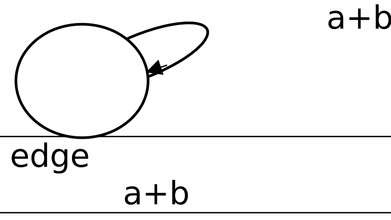
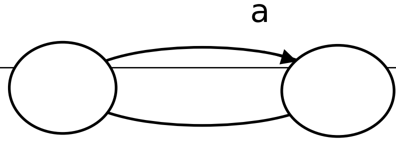
Language	FA	Grammar
L=ab*+ ba*	(i)	(ii)
(iii)		(iv)

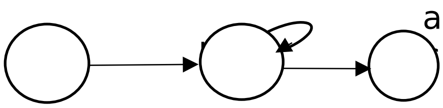
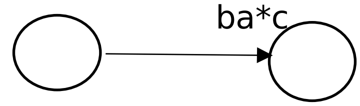
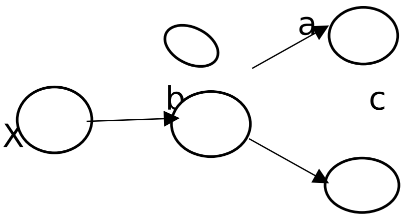
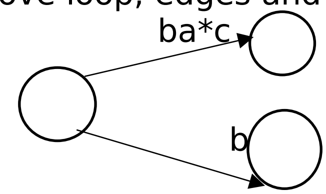
		
(v)	(vi)	A→bX, A→aA, A→λ X→aB, X→λ B→bB, B→λ

Solutions.

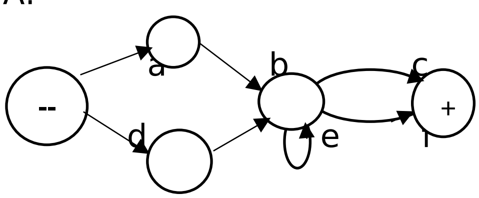
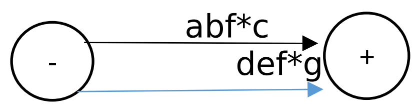
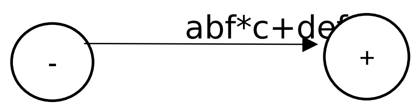
(i) 	(ii) X→aB, X→bA B→bB, B→λ A→aA, A→λ
(iii) L1=a A is final L2=aba* + ba*, B is final, 2 ways to get to B L=L1+L2= a+aba*+ ba*	(iv) X→aA, X→bB A→bB, A→λ B→aB, B→λ
(vi)A is final: from A to A ,L1=a* X is final: from A to X, L2=a*b B is final: from A to B,L3=a*bba* L=L1+L2+L3=a*+a*b+a*bba*	(v)First find FA and then use FA to write the language. FA has 3 final states: A, X, B 

The following are the same. Use them when you want to simplify an FAs

FA	Is Equivalent to this FA
	remove a loop 
a 	remove an edge a+b

b	
	Remove null state 
	remove loop, edges and state Y 

Example. Simplify and find the language of this FA

FA: 	Language: i. Remove null states and some edges  ii. Remove an edge  iii. Hence, the language is : L=abf*c + def*g

Computer Science 323 Names……………………………………………….row#………

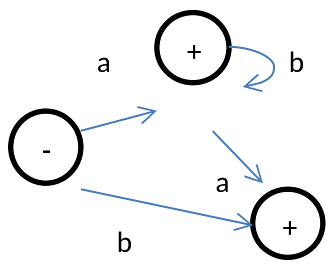
Assignment No.1 …………………… ……………………….

……. …………………………………….

1. True or false? Circle your answers

i.	L=a*ba*	ab is a member of L ba is a member of L lambda is a member of L	True/false True/false True/false
ii.	L=a*b*	a^3b^2 is a member of L b^4 is a member of L	True/false True/false
iii.	L=a* + b*	a^4b is a member of L b^5 is a member of L	True/false true/false
iv.	L=(a* + b)*	a is a member of L bab is a member of L	True/false True/false
v.	L=(ab)*a	$a(ba)^3$ is a member of L abab is a member of L	True/false True/false
vi.	L=a(aa)*(λ+a)b	a*b is a member of L aab is a member of L	True/false True/false
vii.	L=(a+b)*(aa+bb)	aaa is a member of L aabb is a member of L	True/false True/false
viii.	L=(aa)*(λ+a)	a^4 is a member of L a^7 is a member of L L=a*	true/false true/false true/false

2. Complete the following table. Write your answer in each box

Language	FA	CFG
L= c*(a+b)b*		
		
		S → aS |bB B→bB |aA A→aA | bA | λ

3. Find the language of each CFG. Write your answers in the space provided

i. S→aA | bB
A→aA | λ

B→bB | λ

Answer ……………………

ii S→aS | bX | λ

X→aX | bX | λ

Answer …………………………….

4. Programming assignment

Write a program to read a postfix expression and display its numeric value. Suppose a=5,b=7,c=2,d=4

Sample Input/Output

```
Enter a postfix expression with $ at the end: ab+cd*+$
        Value = 20
CONTINUE(y/n)? y
Enter a postfix expression with $ at the end:abcd+++$
        Value = 18
CONTINUE(y/n)? y
Enter a postfix expression with $ at the end:abcd*-*$
        Value = -5
CONTINUE(y/n)? n
```

Directions. Include the following information at the beginning of your program

```
//-----------------------------------------------------------------------------------------------------
//          Group names: Smith, John and Brown, Anna
//          Assignment    : No.1
//          Due date       : …………
// Purpose: this program reads an expression in postfix form, evaluates the expression
// and displays its value
//---------------------------------------------------------------------------------------------------------
```

Comment all functions and class members.