1. **Consider the following grammar G[S]:**

   S→aSbS | aS | c

   Is this grammar ambiguous? if so, please give your reason with an example string and its parse trees.

2. **Consider the following grammar G[S]:**

   S → S0S | S1S | a

   Is this grammar ambiguous? if so, please give your reason with an example string and its parse trees.

3. **Consider the following grammar :**

   G[S]: S→(L) | aS | a

   L→L,S | S

   Please write the rightmost derivation for the sentential form '(S, (a))', and give the handle and the viable prefixes of this sentential form.

4. **Consider the following grammar :**

   G[S]: S→aAcB | Bd

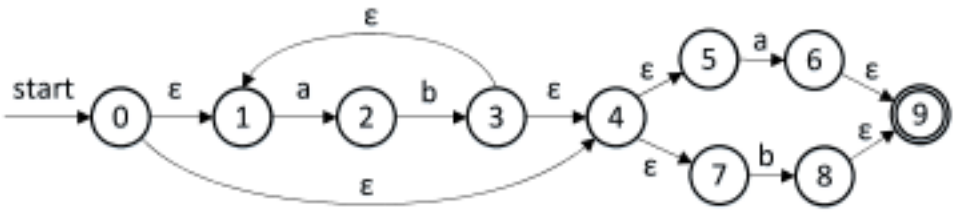
   A→AaB | c

   B→bScA | b

   Please write the rightmost derivation for the sentential form 'aAcbBdcc', and give the handle and the viable prefixes of this sentential form.

5. **Construct the minimum-state DFA for the following NFA:**

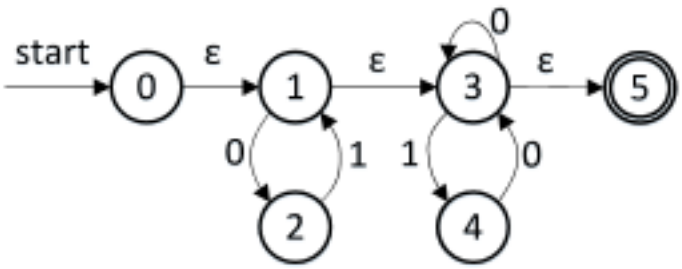
   1) **Convert this NFA into DFA by subset construction. Both the transition table and the transition graph of DFA are required.**
   2) **Minimize the states of this DFA.**



6. Construct the minimum-state DFA for the following NFA:

1) Convert this NFA into DFA by subset construction. Both the transition table and the transition graph of DFA are required.

2) Minimize the states of this DFA.



7. Consider the following grammar G[S]:

S→ **k** | **(** T **)**

T→ T * S | T / S | S

(1) **Please rewrite this grammar to eliminate left recursion.**

(2) **Compute FIRST and FOLLOW for the grammar. Please explain the rewritten grammar is LL(1) grammar or not.**

(3) **Construct the parsing table**

8. Consider the grammar (decls, decl, type, varlist and varlist' are non-terminals):

decls → decl;decls | ε decl → type varlist

type → int | bool varlist → id varlist'

varlist' → , varlist | ε

1) **Construct First and Follow sets for the nonterminals.**

2) **Is this grammar the LL(1) grammar? Give your reason.**

3) **Construct the LL(1) parsing table.**

9. Given the grammar G[S]: S→(S)A | aA , A→ BA | ε , B→ S | +S | *. (here, A→BA having higher priority on A→ ε), and the parsing table of G[S] as follow.

	a	(	)	+	*	$
S	S→aA	S→(S)A				
A	A→ ε A→BA	A→ ε A→BA	A→ ε	A→ ε A→BA	A→ ε A→BA	A→ ε
B	B→S	B→S		B→+S	B→*	

To achieve the predictive parsing, the rule "A→BA having higher priority on A→ε when selecting a production of A for derivations" is added into the grammar G[S]. Please give the parsing process for the input string a*a in the following table.

Step	Stack	Input	Action
1	$S	a*a$	

10.Given the grammar G[S]: S→(S)A | aA , A→ BA | ε , B→ S | +S | *. (here, A→BA having higher priority on A→ ε), and the parsing table of G[S] as follow.

	a	(	)	+	*	$
S	S→aA	S→(S)A				
A	A→ ε A→BA	A→ ε A→BA	A→ ε	A→ ε A→BA	A→ ε A→BA	A→ ε

B	B→S	B→S		B→+S	B→*	

To achieve the predictive parsing, the rule "A→BA having higher priority on A→ε when selecting a production of A for derivations" is added into the grammar G[S]. Please give the parsing process for the input string (a*a) in the following table.

[Answer]:

Step	Stack	Input	Action
1	$S	(a*a)$	

11.Consider the following augmented grammar G[S']:

(0) S'→S (1) S→Pa (2) S→Pb (3) S→c

(4) P→Pd (5) P→Se (6) P→f

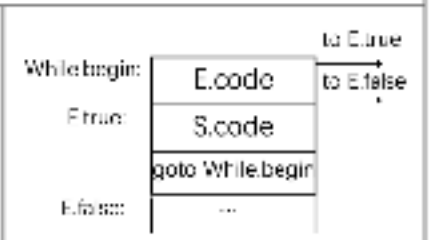
1) **Construct the DFA of LR(0) items for this augmented grammar.**
2) **Is this grammar the LR(0) or SLR(1) grammar ? Give your reason.**
3) **Construct the SLR(1) parsing table.**

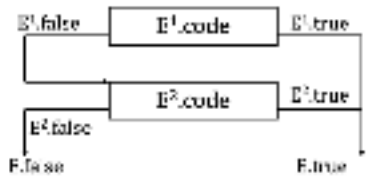
12. Consider the following augmented grammar G[S']:

(0) S'→S (1) S→iDeD (2) S→ iD (3) D→Sb (4) D→ε

1) **Construct the DFA of LR(0) items for this augmented grammar.**
2) **Is this grammar the LR(0) or SLR(1) grammar ? Give your reason.**
3) **Construct the SLR(1) parsing table.**

13. Consider the following attribute grammar:

Grammar	Semantic Rules	
While-stmt→ while E do S	While-stmt.begin=newlabel; While-stmt.next=newlabel; E.true=newlabel; E.false= While-stmt.next; S.next= While-stmt.begin; While-stmt.code=Label While-stmt.begin ‖ E.code ‖ Label E.true ‖ S.code ‖goto While-stmt.begin‖Label While-stmt.next	

E → E1 or E2	E1.true=E.true; E1.false=newlabel; E2.true=E.true; E2.false=E.false; E.code=E1.code ‖ Label E1.false ‖ E2.code	
E → id1 relop id2	E.code = if id1.name relop id2.name goto E.true ‖ goto E.false	
S → id = id+num	S.code = id.name = id.name + num.val	

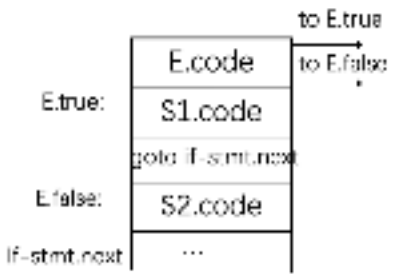
Given the source code: while a<b or c<d do t= t+a

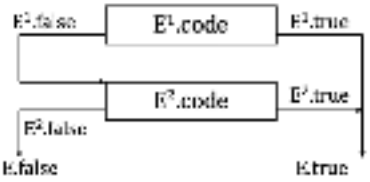
(1) Draw the Abstract Syntax Tree

(2) According to the semantic rules, calculate the inherited attributes 'true' ,'false' and 'next' on the corresponding nodes of the syntax tree, to form the semantic tree。

(3) Consider the step (2) result and the synthetic attribute 'code' , translate the three address code in a bottom-up order, recursively。

14. Consider the following attribute grammar:

Grammar	Semantic Rules	
if-stmt→ if E then S1 else S2	If-stmt.next=newlabel; E.true=newlabel; E.false=newlabel; S1.next=if-stmt.next; S2.next=if-stmt.next If-stmt.code=E.code ‖ Label E.true ‖ S1.code ‖goto S.next ‖ Label E.false ‖ S2.code ‖ if-stmt.next	

E → E1 or E2	E1.true=E.true; E1.false=newlabel; E2.true=E.true; E2.false=E.false; E.code=E1.code || Label E1.false || E2.code	
E → id1 relop id2	E.code = if id1.name relop id2.name goto E.true || goto E.false	
S → id = num	S.code = id.name = num.val	

Given the source code: if a<b or c<d then t= 5 else t=10

(1) **Draw the Abstract Syntax Tree**

(2) **According to the semantic rules, calculate the inherited attributes 'true' ,'false' and 'next' on the corresponding nodes of the syntax tree, to form the semantic tree。**

(3) **Consider the step (2) result and the synthetic attribute 'code' ， translate the three address code in a bottom- up order, recursively。**

考试中有选择题型 20 分。这部分不出复习题了