81. * Symbol table is an important data structure created & maintained by compilers in order to store info about occurrence of various entities such as variable name, function, objects, classes etc.

→ It stores:
- → Associated attributes with identifiers
- → Identifiers & attributes entered by analysis phase
- → < symbol name, type, attribute >

→ It can be implemented using:

① linear list:
- Simplest to implement
- Implemented as array or linked list
- Fast insertion but slow lookup

② Binary Search Tree:
- Can grow dynamically
- Insertion & lookup are O(log n)

③ Hash table:
- Possibility of collisions
- Avg time complexity O(n)

18DCS007

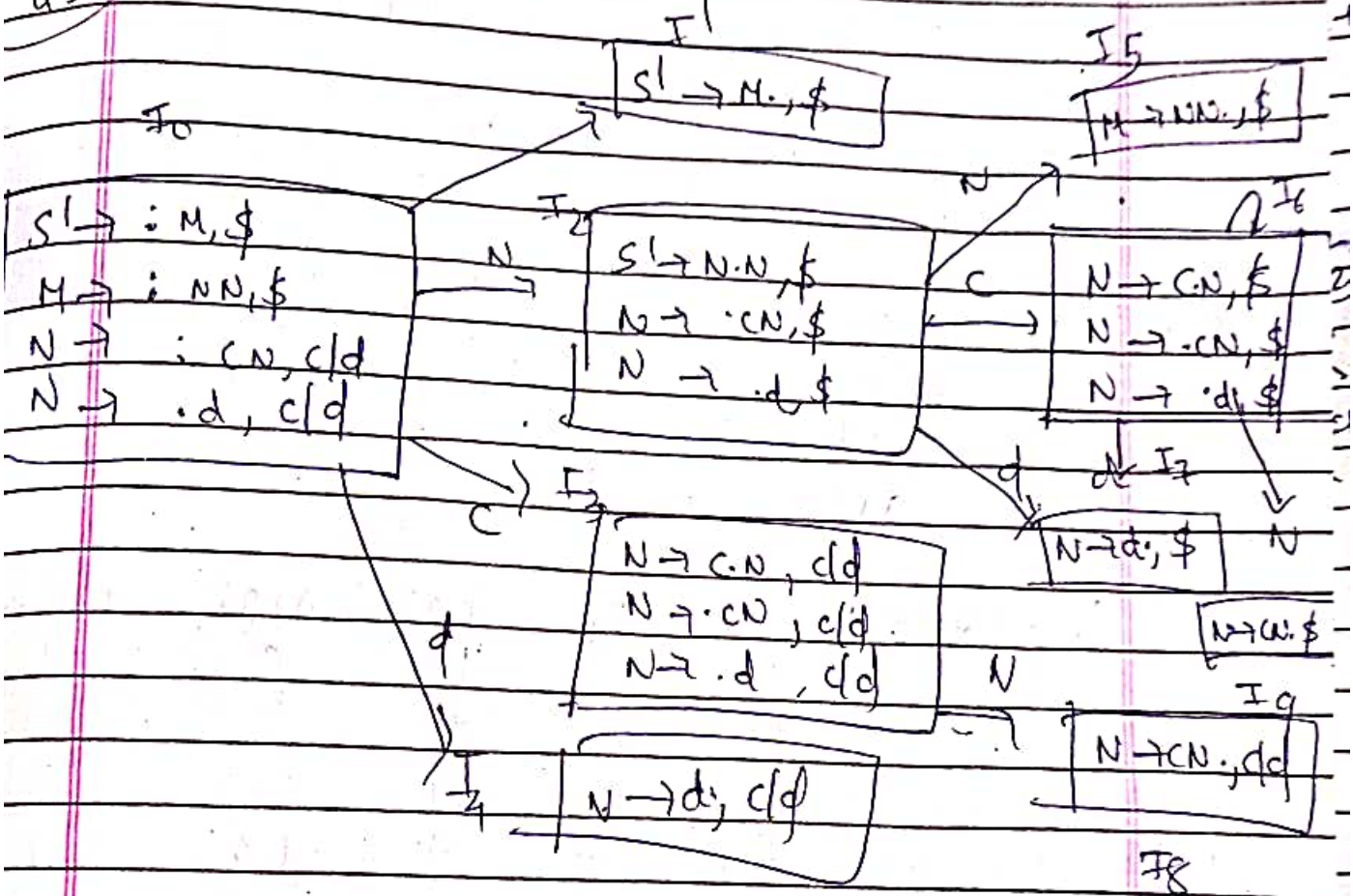
DATE
PAGE NO.

Q2

First (E) = { (, int }
First (T) = { (, int }
First (E') = { +, ^ }
First (F) = { (, int }
First (T') = { ^, * }

Follow (E) = {), $ }
Follow (T) = { +, ^, $ }
Follow (E') = {), $ }
Follow (F) = { ^, +, $ }
Follow (T') = { +, ^, $ }

Q3.



Parsing Table:

	c	d	$	M	N
				1	2
I_0	S3	S4			
I_1			Accept		5
I_2	S6	S7			8
I_3	S3	S4			
I_4	R3	R3			
I_5			R1		
I_6	S6	S7			9
I_7			R3		
I_8	R2	R2			
I_9			R2		

(Q4.) $t_1 = b * c$
$t_2 = d \wedge e$
$t_3 = t_1 * t_2$
$t_4 = a + t_3$

Quadruple :

#	Op	Arg1	Arg2	Result
(0)	*	b	c	t1
(1)	^	d	e	t2
(2)	*	t1	t2	t3
(3)	+	a	t3	t4

Triple:

#	Op	Arg1	Arg2
(0)	*	b	c
(1)	^	d	e
(2)	*	(0)	(1)
(3)	+	(a)	(2)

18DCS007

Indirect Triple:

tt	op	Arg1	Arg2
(14)	*	b	c
(15)	^	d	e
(16)	*	(14)	(15)
(17)	+	a	(16)

List of pointers to table:

tt	statement
(0)	(14)
(1)	(15)
(2)	(16)
(3)	(17)

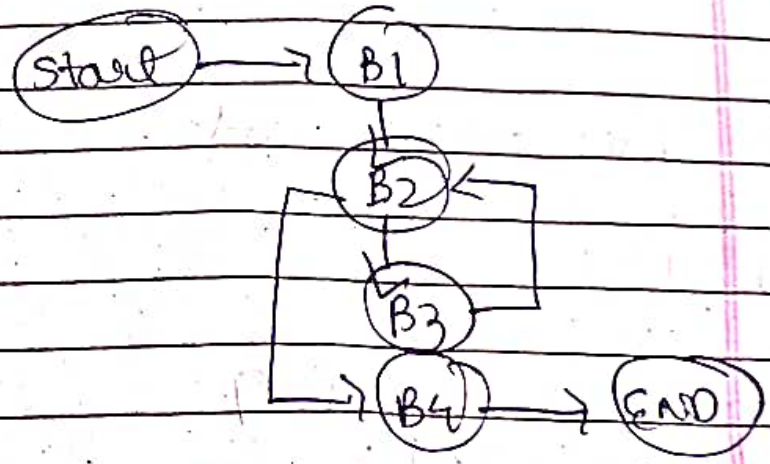
Q5. Three address code:

```
1.  F=1;
2.  i=2;
3.  if (i>a) go to 9
4.  t1 = f * i;
5.  F= t1 ;
6.  t2 = i+1;
7.  i = t2 ;
8.  goto (3)
9.  goto calling program
```

Basic Blocks :

B1 : f = 1 ;
i = 2 ;
B2 : if (i > n) goto 9 *
B3 : t1 = f * i ;
f = t2 ;
t2 = i + 1 ;
i = t2 ;
goto (B2)
B4 : goto calling program *

* Control flow :

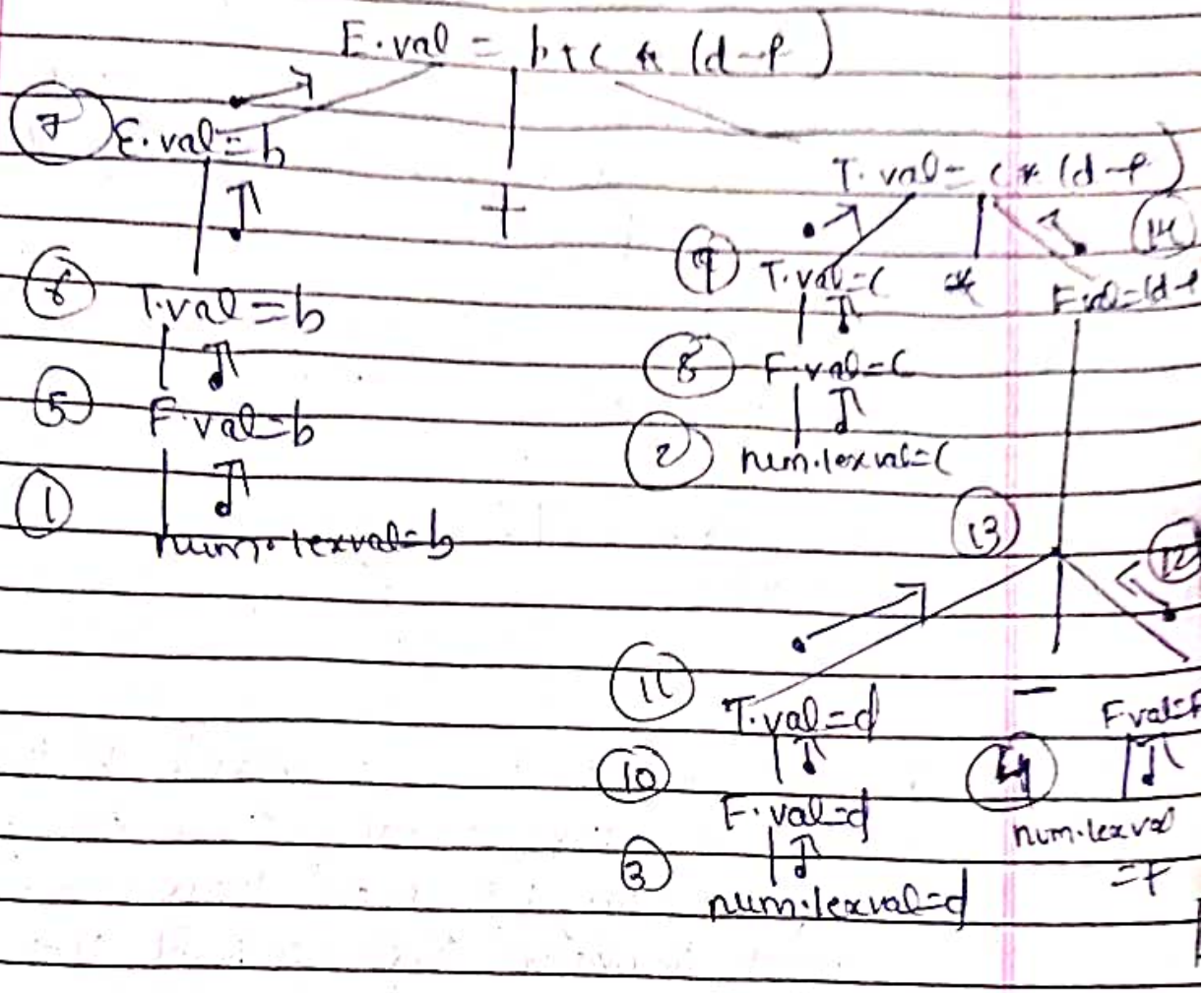


Q6. Syntax Direction Translation :

	Semantic Rules
E → E + T	E.val := E.val + T.v
E → T	E.val := T.val
T → T * F	T.val := T.val *
T → F	T.val := F.val
F → (F)	F.val := F.val
F → T − F	F.val := T.val −
F → num	F.val := num.

DATE
PAGE NO

* Annotated Parse Tree : & Dependency Graph



(Q7) Issues in Code generator :

→ Input to code generator : It works on assumption that input are free from syntactic & state semantic errors.

→ Target program : It is output of code generator. We need additional step after code generator to generate assembly language.

→ Memory Management : A name in three add statement refers to symbol table entry for name.

→ Instruction Selection : Selecting the best instructions will improve the efficiency of the program.

Q8. $E \to TE'$
$E' \to +TE' \mid \varepsilon$
$T \to FT'$
$T' \to T' \to *FT' \mid \varepsilon$
$F \to id$

Q9 $S \to bSS'$
$S' \to aS \mid \varepsilon b$
$S'' \to SaS \mid b \mid c$

Q10 A CFG is said to ambiguous if there exists more than one derivation tree for given input string i.e more than one. left most Derivation Tree or Right Most Derivation Tree.

→ Given Expression : $E \to E+F \mid E*E \mid id$ is ambiguous.

→ We can rewrote the grammer, starting with lowest precedence.

$E \to E+E \mid T$
$T \to T*T \mid F$
$f \to id$