B S SHREESHA
1JT21IS010

21CS51: AUTOMATA THEORY AND COMPILER DESIGN ASSIGNMENT-2

1. Obtain a grammar to generate the following languages

a. $L = \{a^{n+2} b^m \mid n \geq 0 \text{ and } m > n\}$

b. $L = \{a^n b^{2n} \mid n \geq 0\}$

Answer a. $L = \{a^{n+2} b^m \mid n \geq 0 \text{ and } m > n\}$

Strings that can be generated can be represented as

	$n=0$	$n=1$	$n=2$
	$m>0$	$m>1$	$m>2$
i.e.	$m \geq 1$	$m \geq 2$	$m \geq 3$

$L = \{aabb^*, aaabbb^*, aaaabbbb^* \ldots\}$

This is in the form of $a^n b^n \mid n \geq 1$. One a followed zero or more b's. The production can be written as

$$S \longrightarrow aAB$$

where the language $a^n b^n \mid n \geq 1$ is generated from variable A & zero or more b's are generated from variable B. Now we need to write the productions for A & B, which can be written as:

$$A \longrightarrow aAb \mid ab$$
$$B \longrightarrow bB \mid \varepsilon$$

Final grammar $G = (V, T, P, S)$

$$S \longrightarrow aAB$$
$$A \longrightarrow aAb \mid ab$$
$$B \longrightarrow bB \mid \varepsilon$$

4 tuple system:- $V = \{S, A, B\}$

$T = \{a, b\}$

$$P = \left\{ \begin{array}{l} S \rightarrow aAB \\ A \rightarrow aAb \mid ab \\ B \rightarrow bB \mid \varepsilon \end{array} \right\}$$

$S = \{S\}$.

b. $L = \{a^n b^{2n} \mid n \geq 0\}$

The language tells that for every a, there needs to be two b's. This is achieved by adding one or more b at the end.

$S \rightarrow \varepsilon$

$S \rightarrow aSbb$

Final grammar can be written as

$S \rightarrow \varepsilon \mid aSbb$

$G = (V, T, P, S) \Rightarrow V = \{S\}$

$T = \{a, b\}$

$P = \{S \rightarrow \varepsilon \mid aSbb\}$

$S = \{S\}.$

2. Given the following grammar

$E \rightarrow I \mid E+E \mid E^*E \mid (E)$

$I \rightarrow a \mid b \mid c \mid d$

Obtain the derivation tree for the strings $(a+b)^*c^*d$ & $a+b^*c$ and parse tree for each derivation.

Answer $(a+b)^*c^*d$

Leftmost derivation: $E \rightarrow E^*E$

$E \rightarrow E^*E^*E$

$E \rightarrow (E)^*E^*E$

$E \rightarrow (E+E)^*E^*E$

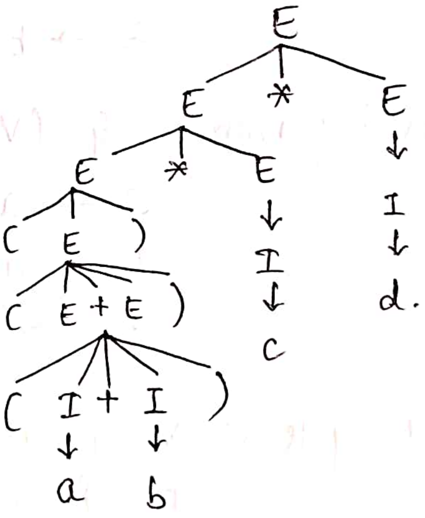
$E \rightarrow (I+E)^*E^*E$

$E \rightarrow (a+I)^*E^*E$

$E \rightarrow (a+b)^*I^*E$

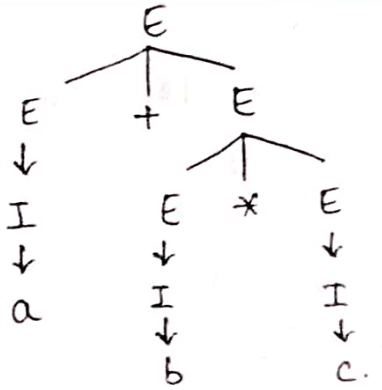
$E \rightarrow (a+b)^*c^*I$

$E \rightarrow (a+b)^*c^*d$



a+ b*c.

Leftmost derivation: E → E+E
E → I+E
E → a+E*E
E → a+I*E
E → a+b*I
E → a+b*c



3. Eliminate all epsilon productions from the grammar

S → BAAB
A → 0A2 | 2A0 | ε
B → AB | 1B | ε

Answer

Old variable (OV)	New variable (nv)	Productions
∅	A, B	A → ε B → ε
A, B	A, B, S	S → BAAB
A, B, S	A, B, S	

Productions	Resulting productions
S → BAAB	S → BAAB | AAB | BAB | BAA | AAB | AB | BA | BB | AA | B | A
A → 0A2	A → 0A2 | 02
A → 2A0	A → 2A0 | 20
B → AB	B → AB | A | B
B → 1B	B → 1B | 1

Final Grammar.

$S \to BAAB \mid AAB \mid BAB \mid BAA \mid AAB \mid AB \mid BA \mid BB \mid AA \mid B \mid A$

$A \to 0A2 \mid 02$

$A \to 2A0 \mid 20$

$B \to AB \mid B \mid A$

$B \to 1B \mid 1$

$G = (V, T, P, S) \Rightarrow V = \{S, A, B\}$

$T = \{0, 1, 2\}$

$S = \{S\}$

$$P = \left\{\begin{array}{l} S \to BAAB \mid AAB \mid BAB \mid BAA \mid AAB \mid AB \mid BA \mid BB \mid AA \mid B \mid A \\ A \to 0A2 \mid 02 \\ A \to 2A0 \mid 20 \\ B \to AB \mid B \mid A \\ B \to 1B \mid 1 \end{array}\right\}$$

4. Design a PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$

Answer Languages accepted :- $L = \{abb, aabbbb \dots\}$.

Transitions.

$\delta(q_0, a, \varepsilon) = (q_0, a)$

$\delta(q_0, a, a) = (q_0, aa)$

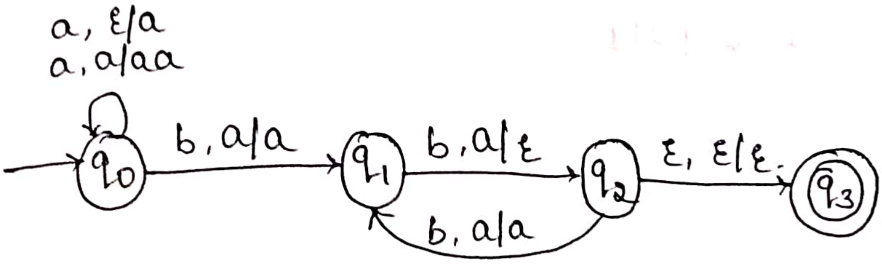
$\delta(q_0, b, a) = (q_1, aa)$

$\delta(q_1, b, a) = (q_2, a)$

$\delta(q_2, b, a) = (q_1, a)$

$\delta(q_1, b, a) = \delta \to$ leads to $\varepsilon \Rightarrow (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, \varepsilon) = (q_3, \varepsilon)$.



To accept

String:- abb.

$\delta(q_0, a, \varepsilon) \vdash (q_0, a)$

$\delta(q_0, a, b) \vdash (q_1, a)$

$\delta(q_1, b, a) \vdash (q_2, \varepsilon)$

$\delta(q_2, \varepsilon, \varepsilon) \vdash (q_3, \varepsilon)$

Accept string

To reject

String:- ab.

$\delta(q_0, a, \varepsilon) \vdash (q_0, a)$

$\delta(q_0, b, a) \vdash (q_1, a)$

There are no more b's to pop a.
Therefore we reject this string

5. Design a turing machine to accept the following language

$$L = \{0^n 1^n \mid n \geq 1\}.$$

Answer consider the following example

XX00YY11.
↑q_0

Step1:- In state q_0, replace 0 by X. change the state to q_1. Move to right

$\delta(q_0, 0) = (q_1, X, R) \Rightarrow$ XXX0YY11
↑q_1

Step2:- Replace 0 by 0 or Y by Y and do not change the state- Move to right

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, Y) = (q_1, Y, R) \Rightarrow$ XXX0YY11
↑q_1

Step3:- If input is 1, replace 1 by Y, change state to q_2. Move to left

$\delta(q_1, 1) = (q_2, Y, L) \Rightarrow$ XXX0YYY1
↑
q_2

Step4:- Replace Y by 0, 0 by 0 & move pointer to left

$\delta(q_2, Y) = (q_2, Y, L)$

$\delta(q_2, 0) = (q_2, 0, L) \Rightarrow$ XXX0YYY1
↑q_2

Step5:- Replace X by X. Change state to q_0, Move to right

$\delta(q_2, X) = (q_0, X, R) \rightarrow$ XXX0YYY1
↑
q_0.

Repeating the steps from 1 to 5, the resulting expression is

XXXXYYYY
↑
q_0.

Step6:- If the scanned symbol is 'y' after X, then there are no 0's left. If there are no 0's, there are no 1's. Change the state to q_3. Replace Y by Y & move pointer to right

$\delta(q_0, Y) = (q_3, Y, R) \Rightarrow$ XXXXYYYY
↑q_3.

If there are any 1's, it can be replaced by using 1-5 steps.

$\delta(q_3, Y) = (q_3, Y, R)$

XXXXYYYYB
↑q_3.

B indicates a blank state which is the end of the string

$\delta(q_3, B) = (q_4, B, R) \Rightarrow$ XXXXYYYYBB
↑q_4.

$M = (Q, \Sigma, \tau, \delta, q_0, B, F) \Rightarrow$

$Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1\}$

$\tau = \{0, 1, X, Y, B\}$

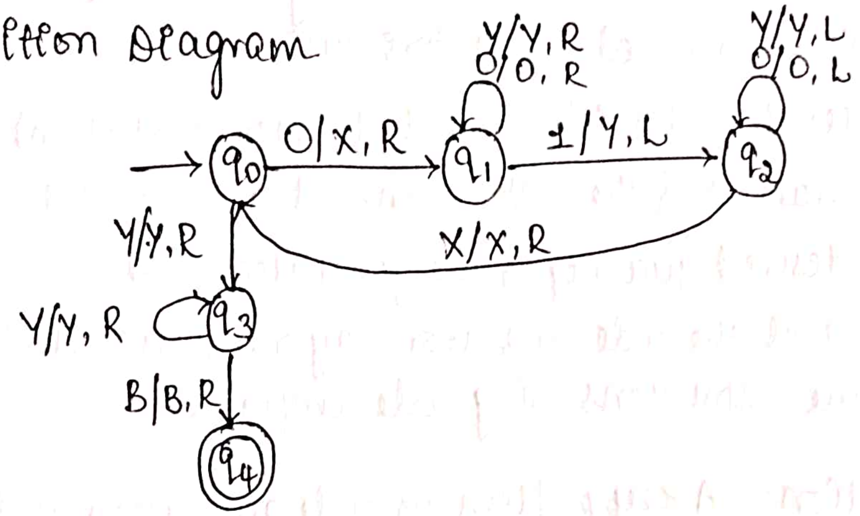
$q_0 = \{q_0\}$

$B = \{B\}$

$F = \{q_4\}$.

δ	Tape Symbols (τ)				
States	0	1	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, 0, R)$	(q_2, Y, L)		(q_1, Y, L)	
q_2	$(q_2, 0, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Transition Diagram



6. Discuss various issues in the design of code generator.

Answer → Input to the Code Generator: The input to the code generator is the intermediate representation of the source program produced by the front-end, along with information in the symbol table that is used to determine the address.

The many choices for the IR include 3 address representations, which include postfix includes, bytecodes etc.

→ The target Program: The instruction set architecture of the target machine has a significance/impact on the difficulty of a good code generator that produces high quality machine code. The most common architectures are RISC (Reduced Instruction Set Computer), CISC (Complex Instruction Set (Architecture)/computer), and stack based

RISC machine has many registers, three-address instructions, simple addressing modes & simple instruction set. In contrast CISC has few registers, two address instructions, variety of addressing modes & instruction side effects.

In a stack-based machine, operations are done by pushing operands onto a stack & then performing the operations on the operands at the top of the stack.

→ Instruction Selection:- The code generator must map the IR program into a code sequence that can be executed by the target machine. The complexity is determined by factors such as

- The level of IR (Intermediate Representation).
- The nature of the instruction-set architecture
- The desired quality of the generated code

If the IR is high level, the code generator may translate each IR into a sequence of machine instructions using code templates.

→ Register Allocation:- A key problem in code generation is deciding what values to hold in what registers. Registers are the fastest computational units on target machines.

The use of registers is often divided into 2 subproblems

- Register allocation, during which we select the set of variables that will reside in registers at each point in the program
- Register assignment, during which we pick the specific register that a variable will reside in.

Finding an optimal assignment of registers to variables is difficult even with single-register machine. Mathematically, the problem is NP-complete.

7. Define shift-reduce parser. Explain its action & conflict by taking an example

Answer Shift-reduce parser is a form of bottom up parsing in which a stack holds grammar symbols and an input symbol is held by the buffer to be parsed. The parse tree is constructed from leaf to the root

ACTION :- The ACTION function takes arguments as state i and a terminal a (or $, the input end marker)

The value of ACTION [i; a] can have 4 forms

- → Shift - Shifts the input to stack
- → Reduce - Reduce the portion to non-terminal symbol
- → Accept - Accepts the input & finishes parsing
- → Error - Discovers the error & takes corrective actions

CONFLICT :- The CONFLICT in shift-reduce parsing occurs when the parser can't decide whether to shift or reduce (OR) it occurs in a state that requests both shift & reduce action.

Consider the example :- $S \rightarrow TL$
$T \rightarrow int \mid float$
$L \rightarrow L, id \mid id$.

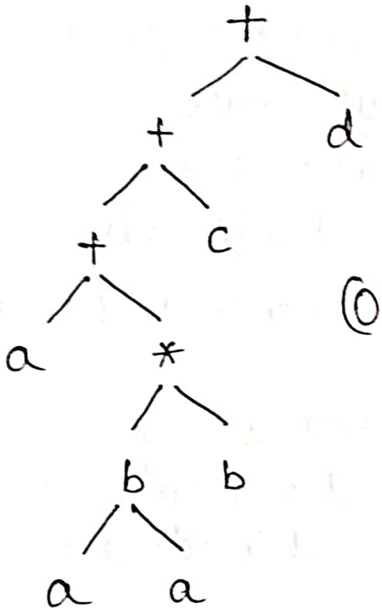
Stack	Input buffer	Parsing Action
$	int id, id; $	shift
$int	id, id; $	Reduce T→int
$T	id, id; $	shift
$T id	, id; $	Reduce L→id
$T L	, id; $	shift
$TL,	id; $	shift
$ TL	; $	Reduce S→TL
$ S	$	Accept.

8. a. Construct DAG for the expression $a+b*(a+b)+c+d$

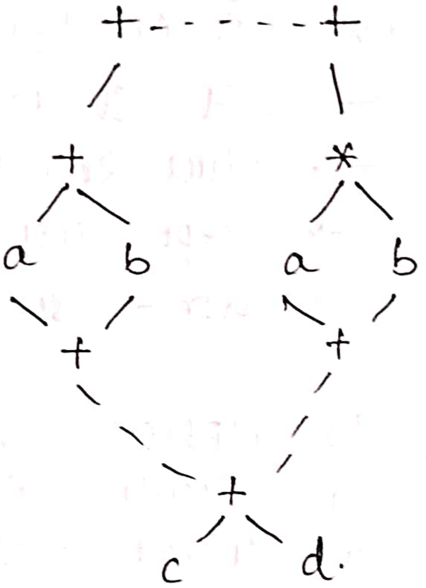
b. Give SDD of simple calculator

Answer a. DAG- Directed Acyclic Graph:- It is a directed graph that does not have any cycles. In DAG, each node represents an operation or a variable and the edges represent the dependencies

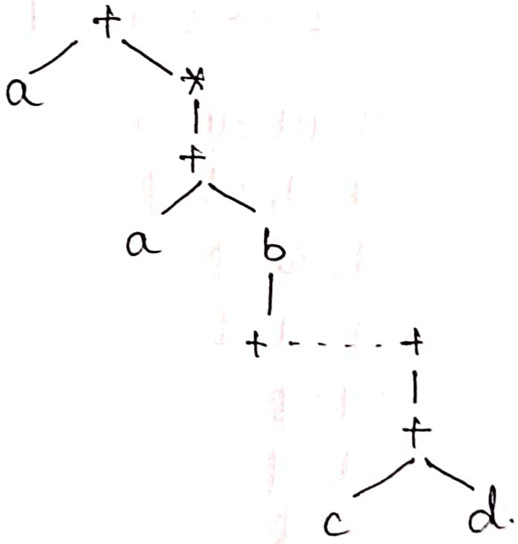
DAG for $a+b*(a+b)+c+d$



(OR)



(OR)

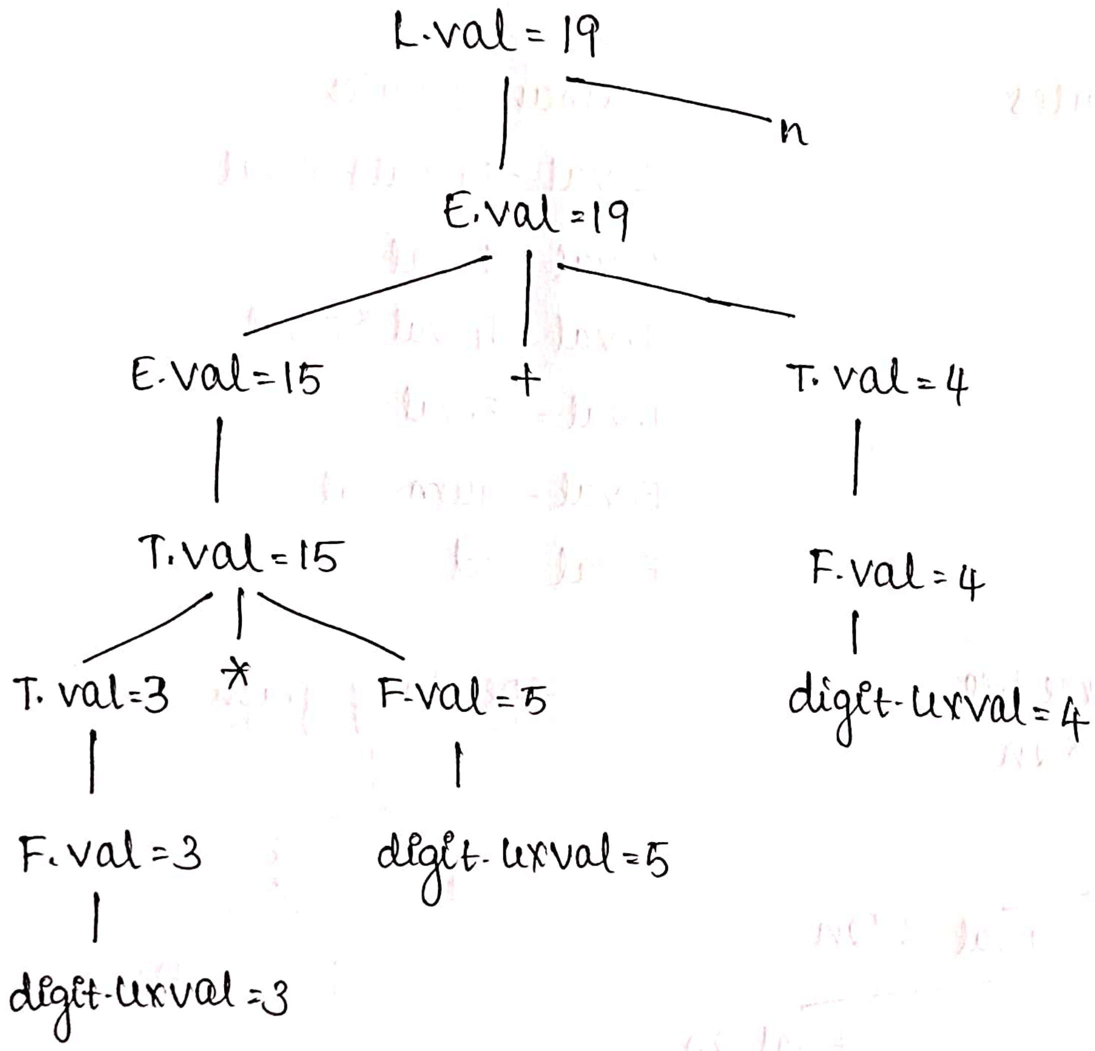


3 different Acyclic Graphs can be drawn.

b.

Production rules	Semantic Rule
$L \rightarrow E\ n$	$L.val = E.val$
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T_1 * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow (E)$	$F.val = E.val$
$F \rightarrow digit$	$F.val = digit.lexval$.

Annotated Parse tree



9. Translate the assignment $a = b^{*} - c + b^{*} - c$ into three address code & quadruples

Answer

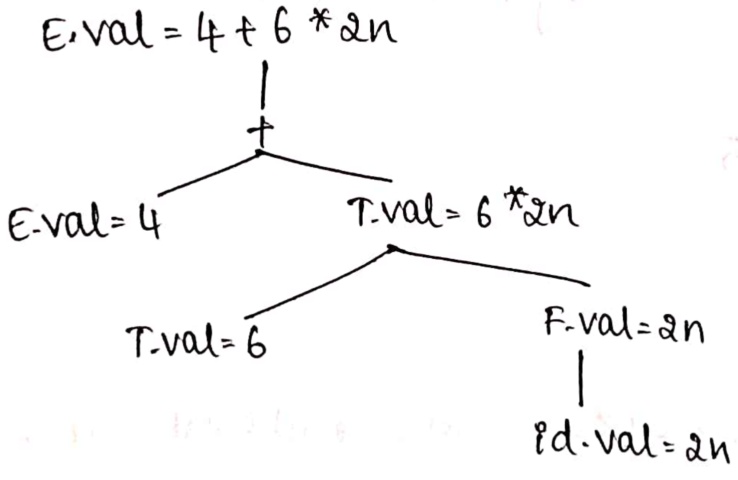
Three Address code	Quadrupes
$t_1 = -c$	$(1, -, c, t_1)$
$t_2 = b * t_1$	$(2, *, b, t_1, t_2)$
$t_3 = -c$	$(3, -, c, t_3)$
$t_4 = b * t_3$	$(4, *, b, t_3, t_4)$
$a = t_2 + t_4$	$(5, +, t_2, t_4, a)$.

10. Give the L-attributed SDD for simple desk calculator & draw the annotation parse tree & dependency graph for the expression 4+6*2n.

Answer

Production rules	Semantic Rules
$E \rightarrow E + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow num$	$F.val = num.val$.
$F \rightarrow id$	$F.val = id$

Annotated parse tree



Dependency graph