Page No.
Date

Q. 4. Define operator precedence grammer? with the help of following given grammer, Parse the input string "a+b*c*d".

$E \rightarrow E+T \mid T$

$T \rightarrow T*V \mid V$

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

→ Operator precedence grammer -

It is a bottom up parsing that interprets an operator precedence grammer.

For example, most calculators use operator precedence parsers to convert from human readable infix notation.

The operator precedence parsing technique can be applied to operator grammers.

$E \rightarrow E+T \mid T$

$T \rightarrow T*V \mid V$

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

	a|b|c|d	+	*	$
a|b|c|d	-	>	>	>
+	<	>	<	>
*	<	>	>	>
$	<	<	<	Reject

Stack	Relation	Input	Action
$ ↑	<	a+b*c*d$ ↑	shift a
$a	>	+b*c*d$	Reduce v→a
$v	<	+b*c*d$	shift +
$v+	<	b*c*d$	shift b
$v+b	>	*c*d$	Reduce v→b
$v+v	<	*c*d$	shift *
$v+v*	<	c*d$	Shift c
$v+v*c	>	*d$	Reduce v→c
$v+v*v	>	*d$	Reduce T→v
$v+v*T	>	*d$	Reduce E→T
$v+v*E	>	*d$	

Input String is not parse using operator precedence grammer.

Page No.
Date

Q.5. Categorize FIRST and FOLLOW set for given grammer below.

$E \rightarrow TE'$
$E' \rightarrow +TE'$
$E' \rightarrow \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT'$
$T' \rightarrow \varepsilon$
$F \rightarrow (E)$
$F \rightarrow id$

→ FIRST sets :-

FIRST (F) = { (, id }
FIRST (T') = { *, ε }
FIRST (T) = { (, id }
FIRST (E') = { +, ε }
FIRST (E) = { (, id }
FIRST (ε) = { ε }
FIRST ((E)) = { (}
FIRST (id) = { id }
FIRST (TE') = { (, id }
FIRST (+TE') = { + }
FIRST (ε) = { ε }
FIRST (FT') = { (, id }
FIRST (*FT') = { * }

FOLLOW set's

FOLLOW (E) = { $,) }
FOLLOW (E) = { $,) }
FOLLOW (T) = { +,), $ }
FOLLOW (T') = { +,), $ }
FOLLOW (F) = { +, *,), $ }

Page No.
Date

Q.5 Diff. bet^n Top-down & Bottom up parsing techniques.

→

Top-down Parsing	Bottom-up parsing.
i) A parsing strategy that first looks at the highest level of parse tree & works down the parse tree by using the rules of formal grammer.	A parsing strategy that first looks at the lowest level of parse tree and works up the parse tree by using the rules of a formal grammer.
ii) Process starts with Root	Process starts with leaves
iii) It starts with starting symbol of grammer	It ends with starting symbol of grammer.
iv) This parsing technique uses Left most Derivation	This parsing technique uses Right most Derivation.
v) It is not accepting Ambiguous Grammer	It is accepting Ambiguous Grammer.
vi) It is less powerful than bottom up parser	It is more powerful than Top down parser.
vii) It is simple to produce parser	It is difficult to produce parser.
viii) It uses LL(1) grammer to perform parsing	It uses SLR, CLR, LALR grammer to perform parsing.
ix) Error detection is weak	Error detection is strong.

Q.7. Define i) macro,
ii) macro expansion

→ i) macro -

~~Macro is a pre processor directive~~

Macro is a sequence of instruction assigned by a name which can be ured at any location in the program.

ii) macro expansion -

Macro expansion is the replacement of macro call into macro definition

Typer :- 1) Lexical expansion
2) Semantic expansion

Page No.
Date

Q.12. Eliminate left recursion from the following grammer.

$S \rightarrow (L) / x$

$L \rightarrow L, S / S$

→ The grammer after eliminating left recursion is

$S \rightarrow (L) / x$

$L \rightarrow SL'$

$L' \rightarrow , SL' / \varepsilon$

Q.4. Define macro call with example.

→

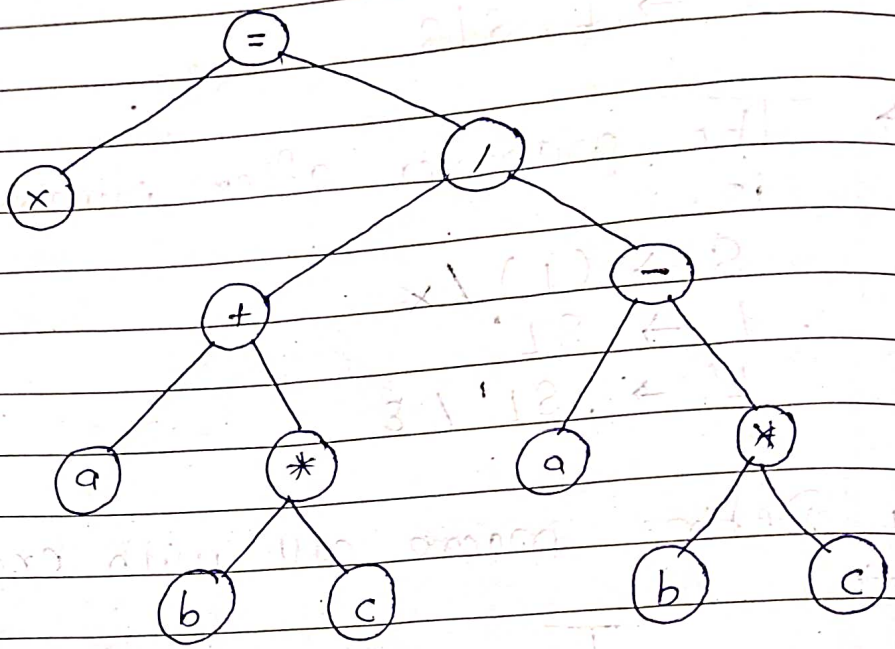
To call the macro, the macro name is written in the mnemonic field of assembly statement.

Syntax

<macro name> [<actual parameter specification> [,...]]

Q.2. Design Syntax tree for $x = (a+b*c)/(a-b*c)$

→



Q.3. Define a context free grammer

→

CFG is a formal grammer which is used to generate all the possible patterns of strings in a given formal languages.

A CFG has 4 tuple

$G = (V_N, V_T, P, S)$

where,

V_N = Set of non-terminal symbol

V_T = Set of terminal symbol.

P = set of production rules.

S = Special non-terminal SEN, which is the Start symbol.

Page No.
Date

Q. 3. Define Finite State Automata.

→ Finite automata is a machine with a finite number of states in which machine can perform.
A finite automata consists of states and transitions, we denote states as circles and transitions as arrows connecting one state to another.

Q. 1. List out the features of macro.

→

1) Some Standard system variable Symbols.
2) conditional assembly
3) The use of one system variable symbol to solve the branch problem.
4) The use of concatenation to generate type-specific instructions.

Page No.
Date

Q.11 Define Top Down Parsing ? State Drawbacks of Top-Down Parsing.

→

i) A parsing strategy that first looks at highest level of parse tree & works down parse tree by using the ruler of formal grammer.
ii) Procers starts with Root.
iii) This parsing technique uses Left most Derivation.

Drawbacks :-

i) In the top-down parsing, each terminal Symbol produces by multiple production of the grammer. (which is predicted) is connected with the input string Symbol pointed by string maker. If match is succesful, the parser can sustain, If the mismatch occurs, then predictions have gone wrong.
ii) Backtracking was major drawback of top down parsing.

Page No.
Date

9.9. List the advantage and disadvantages of operator precedence parsing.

→ Advantages :-

1) It can easily be constructed by hand.
2) It is simple to implement this type of parsing.
3) Simple Powerful enough for expressions in programming languages.

Disadvantages :-

1) Small class of grammers
2) It cannot handle the unary minus.
3) Difficult to decide which language is recognized by the grammer.
4) It is applicable only to small class of grammers.

MACRO Pass-I Flowchart