

Compiler Design

S.No: 25

Peska Sathwik

Analytical Day-1:-

- ① Find a no. of tokens in the following C statement is

```
int main()
```

```
{
```

```
/* find man of a and b */
```

```
int a=10, b=30;
```

```
if (a < b)
```

```
    return (b);
```

```
else
```

```
    return(a);
```

```
}
```

```
int main()
```

```
/* find man of a and b */
```

```
int a=10;
```

```
b=30;
```

```
if (a < b)
```

```
    return (b);
```

```
else
```

```
    return (a);
```

```
}
```

∴ No. of tokens = 31

② Write a regular expression to denote set of all strings over $\{0, 1\}^*$ containing substring "101".

S.No.: 25
P. Sathwik

Expression: $^*(?= *101) \cdot ^* \$$

Explanation:

(i) '^' starting of string

(ii) $(?= *101)$ for checking if "101" anywhere in the string or not

(iii) '*' matches zero or more.

(iv) '\$' asserts the end of string.

String contains the substring "101"

③ Write a regular expression that have at least two consecutive 0's (or) 1's

Expression: $^*(^*00^* \cdot 1 \cdot ^*11 \cdot ^*) \$$

Explanation:

① '^' starting of string

② $(^*00^* \cdot 1 \cdot ^*11 \cdot ^*)$ group that matches consecutive 0 or 1

③ '·' logical or operator

This expression matches any string that atleast two characters.

④ How would you trace the program segment $a = (b+c)* (b*c)$ for all phases of compiler.

S.No: 25

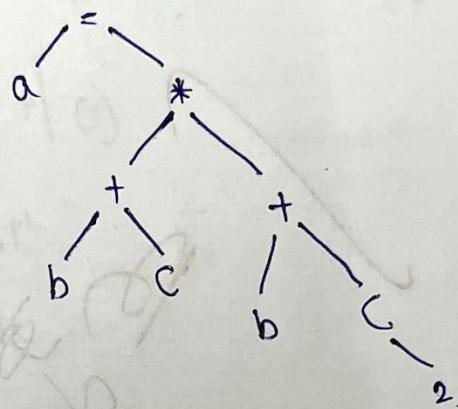
P. Sathwik

Lexical Analysis. Token:

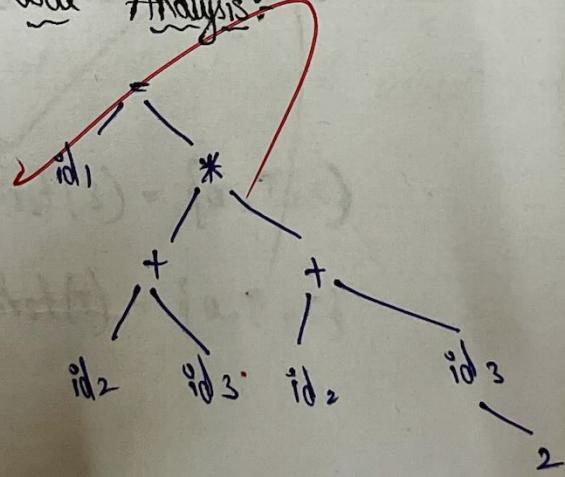
"a", "=", "(", "b", "+", "c", ")", "*", "(-", "b", "+", "c", ")", "*", "2"

d. no. VOT

Syntax Analysis: Parse tree



Semantic Code Analysis:



Intermediate Code generation:-

$$t_1 = b + c$$

$$t_2 = t_1 * t_1$$

$$a = t_2 \ll 1$$

Code Optimization:

The optimized generated code is transferred into machine code.

Code Generator:

MOV eax, b

ADD eax, b

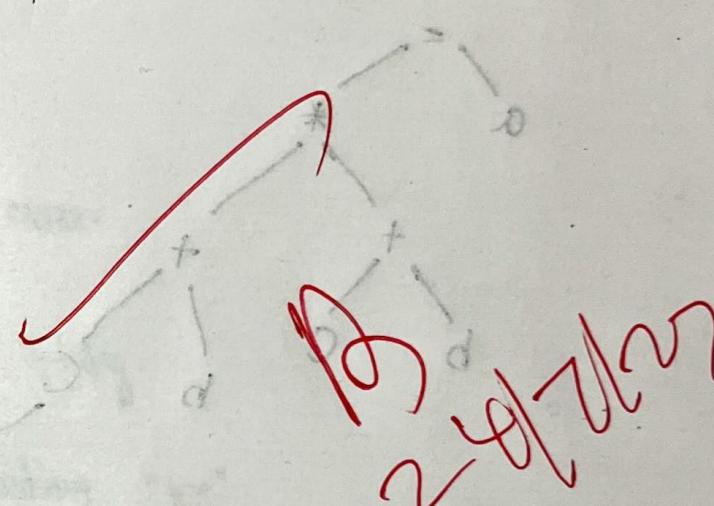
MUL eax, eax

SHL eax, 1

MOV a, eax

S.No: 25

P. Sathwik



Analytical Day - 2

S.No : 25

① Predictive Parser

$$S \rightarrow a \mid \uparrow \mid (T) \quad T \rightarrow T, \quad S \mid s$$

$$(i) \quad S \rightarrow a \mid \uparrow \mid (T)$$

$$T \rightarrow T, \quad S \mid s$$

Q12. ②

Step 01: Eliminate left-recursion from grammar

$$S \rightarrow a \mid \uparrow \mid (T)$$

removing left recursion

: 10 pt 8

$$T \rightarrow ST'$$

$\downarrow \leftarrow b$

$$T \rightarrow S \mid s$$

$\downarrow \leftarrow 2$

Step 02: Create the first and follow sets for each non-terminal

Step ①

$$\text{first}(S) = \{a, T, c\}$$

$$\text{first}(T) = \{a, \uparrow, c\}$$

Step ②

$$\text{follow}(S) = \{\$,)\}$$

$$\text{follow}(T) = \{, \$,)\}$$

Step ③ Predictive Parsing table

(ii)

	a	\uparrow	c	,	\$
S	a	\uparrow	c		

(a) words

$\downarrow \leftarrow b \quad .1$

(b) start

Parsing tab for T:

	a	\uparrow	c	,	:	\$
T	a	\uparrow	c	$\downarrow \downarrow$	$T \leftarrow T$	
T'				.		

S.No: 25
P. Sathwik

②

SLR

$$S \rightarrow CC \quad C \rightarrow CC \mid d$$

Step 01: Augment the grammar

$$S' \rightarrow S$$

$$S \rightarrow CC$$

C \rightarrow CC/d. the wallet bag book etc etc) : opt

Step 02: Compute the closure & go to sets for item.

LR(0) item:

$$1. \quad S' \rightarrow S$$

$$2. \quad S \rightarrow .CC$$

$$3. \quad S \rightarrow .CC$$

$$4. \quad S \rightarrow .d$$

$$5. \quad C \rightarrow .CC$$

$$6. \quad C \rightarrow .d$$

$$\{S', S, d\} = (2) \text{ front}$$

$$\{S, C, d\} = (1) \text{ front}$$

$$\{C, d\} = (2) \text{ wallet}$$

$$\{C, d, \}\text{ - } (1) \text{ wallet}$$

10:

Closure (10)

$$1. \quad S' \rightarrow .S$$

Go to (10):

$$\text{Go To } (10, S) = 11$$

	.	0	1	0	

(ii)

1. $S' \rightarrow S$

Go To (1):

Go To (1, C) = 1₂

1₂: closure (1₂)

1. $S \rightarrow .cc$

2. $S \rightarrow .cc$

3. $S \rightarrow .d$

4. $C \rightarrow .cc$

5. $C \rightarrow .d$

Go To (1₂):

Go To (1₂, C) = 1₃

Go To (1₂, C) = 1₄

Go To (1₂, d) = 1₅

13:
closure (13):

1. $S \rightarrow cc$

2. $C \rightarrow .cc$

3. $C \rightarrow .d$

Go To (13):

Go To (13, C) = 1₆

14:

closure (14):

1. $S \rightarrow c.c$

2. $C \rightarrow .cc$

3. $C \rightarrow .d$

Go To (14):

Go To (14, C) = 1₇

15: closure(15)

closure(0):

8.No:25
P.Sathwik

1. $C \rightarrow CC$

2. $C \rightarrow .CC$

3. $C \rightarrow d$

Go To $(16, C) = 18$

16. Closure(16):

$C \rightarrow CC$

Closure(16)

1. $C \rightarrow d$

	c	d	\$: (c1) S T o o C
10				Shift 1
11				Accept = (S, \$) o T o o
12	shift 3	shift 5		shift 4 • shift 6
13	shift 3	shift 5		"
14	"	"		: (ER) u 9800 B u
15				Reduce 2
16				b. < 2 . S
17				"
18	"	"		: (A) 9800 B u

3A) Grammar

$$S \rightarrow AaAb \mid BbBa$$

$$A \rightarrow \epsilon$$

$B \rightarrow \epsilon$ is LL(1) or not?

$$S \rightarrow AaAb \mid BbBa$$

$$A \rightarrow \epsilon$$

$$B \rightarrow \epsilon$$

Step 01: First step for non-terminal

$$\text{FIRST}(A) = \{\epsilon\}$$

$$\text{FIRST}(B) = \{\epsilon\}$$

$$\text{FIRST}(S) = \{a, b\}$$

Not LL(1)

4A) Grammar

$$E \rightarrow 2E_2$$

$$E \rightarrow 3E_3$$

$$E \rightarrow 4$$

Passing input string 32423

Step 1: Initialization

$$\text{Stack} = \$$$

$$\text{Input: } 32423\$$$

Step 2: Parsing

S.No: 25

P. Sathwik

\$	32423 \$	Shift 3, Push 3	S.No:28 P. Sathwik
\$	2423 \$	Shift 2, Push 2	
\$32	423 \$	Reduce $\Rightarrow 4$	
\$E	4 23 \$	POP 4	
\$E4	23 \$	Shift 4, Push 4	
\$E42	3 \$	Shift 2, Push 2	
\$E4	3 \$	Reduce $E \rightarrow 4$	
\$E	3 \$	POP 4	
\$F3	\$	Reduce $E \rightarrow 2E2$	
\$E3\$	-	Pop E	
\$E	-	Shift 3, Push 3	

shift 3, Push 3

Reduce $F \rightarrow 3E3$

Accept

DO NOT DO IT

SCALE price type price

original or not?

2 = do it

fee see : word

private : single

Analytical Day - 3

S.No: 25

P. Sathwik

① Syntax Directed translation

$2^0 (4+5)$

$E \rightarrow E + T \mid E - + \mid T$

$T \rightarrow T * F \mid T \mid f \mid F$

$F \rightarrow (E) \mid \text{num}$

$E \rightarrow \text{Represent expression}$

$T \rightarrow \text{Term}$

$F \rightarrow \text{Factor.}$

SDT Action:

$E \rightarrow E + T \quad \{ \text{right} = \text{Pop}(); \text{left} = \text{Pop}(); \text{exit}$

$\quad \quad \quad (\text{left} + \text{right}) + (+); \text{Push}(\text{left} + \text{right} + (+)); \}$

$| E - T \quad \{ \text{Right} = \text{Pop}; \text{left} = \text{Pop}; \text{Exist}$

$\quad \quad \quad \text{left} + \text{right} + (-); \text{Push}(\text{left} + \text{right} + (-)); \}$

$| T \quad \{ \text{result} = \text{Pop}(); \text{Push}(\text{result}); \}$

$T \rightarrow T * F \quad \{ \text{right} = \text{Pop}; \text{left} = \text{Pop}; \text{exit} \quad \text{left} + \text{right} + (*); \}$

$\quad \quad \quad \text{Push}(\text{left} + \text{right} + (*)); \}$

$| F \quad \{ \text{result} = \text{Pop}(); \text{push}(\text{result}); \}$

$F \rightarrow (E) \quad \{ \text{result} = \text{Pop}(); \text{push}(\text{result}); \}$

{num { Exit (num); Push (num); }}

CN: 25

② SSD Scheme

$3^* 5 + 6^* 3$

$E \rightarrow E + T \mid E \rightarrow T \mid T$

$T \rightarrow T^* F \mid T \mid F \mid f$

$F \rightarrow (E) \mid \text{num}$

$(2+3)^* 6$

$T \mid T^* F \mid T \mid F \leftarrow T$

$T \mid T \mid T \mid T^* T \leftarrow T$

$\text{num} \mid (3) \leftarrow T$

Semantic rule:

1. $E \rightarrow E_1 + T \quad \{E.\text{Val} = E_1.\text{Val} + T.\text{Val}\}$

$\{E_1 - T \quad \{E.\text{Val} = E_1.\text{Val} - T.\text{Val}\}\}$

$\{T \cdot \{E.\text{Val} = T.\text{Val}\}\}$

2. $F \rightarrow (E) \quad \{E.\text{Val} = E.\text{Val}$

$\mid \text{num} \quad \{\text{num}.\text{Val} = \text{num}.\text{Val}\}$

$\{(1 + 2) * 3 \mid 1 * 2 + 3\} \mid 1 * 2 + 3 \mid 1 * (2 + 3)$

③ Grammar G

$E' \rightarrow E$

$E \rightarrow E + n \mid n$

Parsing Action

Stack

input

\$

$n+n$

Action

$\$n$

$+n$

shift '+'

$\$n+$

n

shift 'n'

$\$nn$

\$

$E \rightarrow n$

P. Sathwik

Reduce $E \rightarrow E + n$

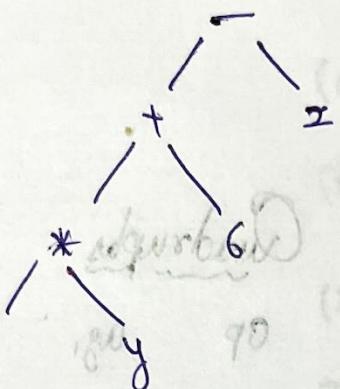
$\frac{1}{2} E_1$

$$\sin(2 + \delta_{10}) - (b +$$

Reduce $E' \rightarrow E$

Tree:

$$x^* y + b = 2$$



⑤ Syntax directed

$S \rightarrow EN$

$$E \rightarrow E + T \quad |E - T| \neq T$$

$$T \rightarrow T^* F / \langle T | F \rangle_F$$

F | F → (E) | digit

670

(10)

501

1

6

input

S^* 3+2

~~26/7/03~~

Translate the expression $-(a+b)*(c+d) + (a+b+c)$ into

(i) quadruples

(ii) Triples

(iii) Indirect triples.

Given $-(a+b)*(c+d) + (a+b+c)$

$$t_1 = a+b$$

$$t_2 = 0 \cdot t_1$$

$$t_3 = c+d$$

$$t_4 = t_2 * t_3$$

$$t_5 = t_1 + c$$

$$t_6 = t_4 + t_5$$

Quadruples

	OP	arg ₁	arg ₂	Result
(0)	+	a	b	t
(1)	-	0	t ₁	t ₂
(2)	+	c	d	t ₃
(3)	*	t ₂	t ₃	t ₄
(4)	+	t ₁	t ₄	t ₅
(5)	+	t ₄	t ₅	t ₆

Triples :-

	OP	arg ₁	arg ₂
(0)	+	a	b
(1)	-	(0)	-
(2)	+	t ₁	d
(3)	*	(0)	(2)
(4)	+	(0)	c
(5)	+	(3)	(4)

Indirect triple :-

- 100 (0)
- 101 (1)
- 102 (2)
- 103 (3)
- 104 (4)
- 105 (5)

② Translate the expression $a^* - (b+c)$

a) Quadruples

b) Postfix Notation

c) Three-address code

b⁴, c⁵, t²
S.No: 25

P. Sathwik
d + e = 50

$$t - d = 9$$

Quadruples:

$$a^* - (b+c)$$

$$t_1 = b+c$$

$$t_2 = 0+1$$

$$t_3 = 0+a$$

$$t_4 = t_3 * t_2$$

	OP	arg ₁	arg ₂	result
(1)	+	b	-	c $t_{20} = t_1$
(2)	-	0	-	1 $t_{21} = t_2$
(3)	+	0	+a	$t_{22} = t_3$
	*	t_{22}	t_{21}	$t_{23} = t_4$

b) Postfix Notation:

$$a^* - (b+c) \Rightarrow abc + - *$$

c) Three Address Code:

$$t_1 = b+c$$

$$t_2 = -t_1$$

$$t_3 = a^* t_2$$

$$\textcircled{3} \quad t_1 = c * d$$

$$t_2 = a + b$$

$$q = t_2 - t_1$$

S.N. No: 25

P. Sathwik

\textcircled{4}

$$t_1 = a * b$$

$$t_2 = c * d$$

$$t_3 = d * e$$

$$t_4 = t_1 \text{ or } t_2$$

$$t_5 = t_4 \text{ and } t_3$$

Back Patching:

if t_4 goto 4

if not t_5 goto t_2

L1:

goto L2:

t_2

L3: 1

27/1/23

Analytical Day-5

① DAG for following three address code:

$$a = b + c$$

$$t_1 = a \times a$$

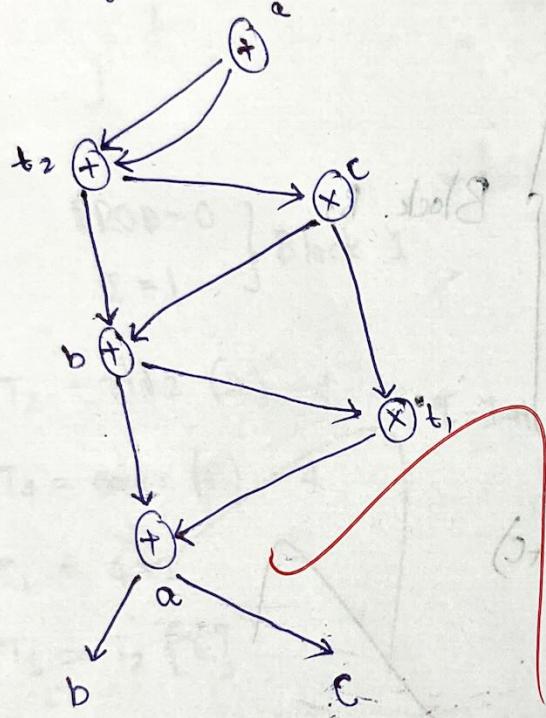
$$b = t_1 + a$$

$$c = t_1 \times b$$

$$t_2 = c + b$$

$$a = t_2 + t_1$$

A) Directed cyclic graph:



② Basic blocks

$$(1) PROD = 0$$

$$(2) I = 1$$

$$(3) T_2 = \text{addr}(A) - 4$$

$$(4) T_4 = \text{addr}(B) - 4$$

$$(5) T_1 = 4 \times 1$$

$$(6) T_3 = T_2(T_1)$$

$$(7) T_B = T_3 \times T_5$$

$$(8) T_6 = T_3 \times T_5$$

$$(9) PROD = PROD + T_6$$

$$(10) I = I + 1$$

$$(11) \text{If } I < 20 \text{ GoTo } (3)$$

0.9 1 0.999

0.9 7 0.999

0.1 1 0.999

0 = 0099

S.No.: 25

P. Sathwick

0 = 0099(1)

I = 1 (2)

$\Delta - (1) \text{ add} \Rightarrow ST (3)$

$\Delta - (2) \text{ mult} = PT (4)$

$1 \times 1 = PT (2)$

$[, PT] ST = ST (3)$

$[, PT] PT = ST (4)$

$ST \times ST = ST (5)$

$PT + 0099 = 0099 (6)$

$PT = PT (6)$

$(2) \text{ add} \Rightarrow PT (7)$

$G(A) + G(A) + (A+A) = W (2)$

$PT = ST$

$PT = ST$

$ST = ST$

$T_6 = T_3 \times T_5$

$PROD = PROD + T_6$

$I = I + 1$

$\text{If } I < 20 \text{ GoTo } (3)$

0.9 1 0.999

0.9 7 0.999

0.1 1 0.999

APP C, R0

Mov R0, W

Basic block and flow graph

for (i=1 to n)

{

s=1;

while (j < n)

{

A=B*(C/D);

j=j+1;

}

Any:

PROD=0 } Block 1
I=1

$$T_2 = \text{add2}(A) - 4$$

$$T_4 = \text{add2}(B) - 4$$

$$T_1 = 4 \times I$$

$$T_3 = T_2[T_1]$$

$$T_5 = T_4[T_1]$$

$$T_6 = T_3 \times T_5$$

$$\text{PROD} = \text{PROD} + T_6$$

I=I+1

$$3 | A \leftarrow A$$

$$3 | d \leftarrow d$$

$$3 | s \leftarrow s$$

Block 2

$$3 | s \leftarrow A$$

27/7/23

Analytical Day-6

28/07/23

i) Consider the following grammar and eliminate left recursion.

$$A \rightarrow ABd / Aa / a$$

$$B \rightarrow Be / b$$

Step①: Identify left-recursion variables:

The left-recursive variables in the grammar are A & B , as they appear as the leftmost symbol in some of their production rules.

Step②: Eliminate left recursion:

$$A \rightarrow aA'$$

$$A' \rightarrow BaA' / \epsilon$$

$$B \rightarrow bB'$$

$$B' \rightarrow eB' / \epsilon$$

(ii) Consider the following grammar & eliminate left recursion

$$A \rightarrow AA \alpha / \beta$$

Ans: Left recursion is eliminated by converting the grammar into a right recursive grammar.

$$A \rightarrow BA'$$

$$A' \rightarrow A\alpha A' / \epsilon$$

Explanation: If α is non-terminal, we create a new non-terminal β to handle the recursion. Now, α can produce strings starting with β . β can produce strings starting with $\alpha\alpha$ or ϵ . This way, we avoid left recursion for α .

S.No: 25

P. Sathwik

(iii) Do left factoring in the following grammar

a. $S \rightarrow bSSaS / bSSaSb / bSb / a$

b. $S \rightarrow aSSbS / aSaSb / abb / b$

a) Step ①: Identify common prefixes

Common prefix in this grammar is "bSS"

Step ②: Perform left-factoring:

Factor out common prefix "bSS" & create new non-terminals to handle the different suffixes:

$$\begin{aligned} S &\rightarrow bSSS' \\ S' &\rightarrow aaS / aSb / b \end{aligned}$$

b) ① Common prefix is "aS"

$$S \rightarrow aSS'S'$$

$$S' \rightarrow bS / aSb / \epsilon$$

iv) Check whether the following grammar is a LL(1) or not

$$S \rightarrow iE + S \mid iE + SeS \mid a$$

$$E \rightarrow b$$

S.No: 25

P. Sathwik

Ans:

$$\text{FIRST}(S) = \{i, a\}$$

$$\text{FIRST}(E) = \{b\}$$

Follow sets:

$$\text{FOLLOW}(S) = \{\$\}$$

$$\text{FOLLOW}(E) = \{i, a, \$\}$$

D 27/07/2023

222d ← 2

3|d20|200 ← 2

12'22d ← 2

3|d20|2d ← 2

Analytical 07

① Can build a recursive descent parser for the following grammar using

$$E \rightarrow TE'$$

$$E' \rightarrow +TE'/\epsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow +FT'/\epsilon$$

$$F \rightarrow (\epsilon) | id$$

S.No: 25

P. Sathwik

Procedure $E()$

```
{
    T();
    E'();
}
```

$$E \rightarrow TE'$$

Procedure $E'()$

```
{
    if input symbol = '+' then
        advance();
    T();
    E'();
}
```

(+) - leading token for

: operator

(+) - leading token for id

: variable

(id)

'(' - leading token for

: variable

(id) no. 2

(id) no. 3

(id) no. 4

(id) no. 5

(id) no. 6

(id) no. 7

(id) no. 8

(id) no. 9

(id) no. 10

(id) no. 11

(id) no. 12

(id) no. 13

(id) no. 14

(id) no. 15

(id) no. 16

(id) no. 17

(id) no. 18

(id) no. 19

(id) no. 20

(id) no. 21

(id) no. 22

(id) no. 23

(id) no. 24

(id) no. 25

(id) no. 26

(id) no. 27

(id) no. 28

(id) no. 29

(id) no. 30

(id) no. 31

(id) no. 32

(id) no. 33

(id) no. 34

(id) no. 35

(id) no. 36

(id) no. 37

(id) no. 38

(id) no. 39

(id) no. 40

(id) no. 41

(id) no. 42

(id) no. 43

(id) no. 44

(id) no. 45

(id) no. 46

(id) no. 47

(id) no. 48

(id) no. 49

(id) no. 50

(id) no. 51

(id) no. 52

(id) no. 53

(id) no. 54

(id) no. 55

(id) no. 56

(id) no. 57

(id) no. 58

(id) no. 59

(id) no. 60

(id) no. 61

(id) no. 62

(id) no. 63

(id) no. 64

(id) no. 65

(id) no. 66

(id) no. 67

(id) no. 68

(id) no. 69

(id) no. 70

(id) no. 71

(id) no. 72

(id) no. 73

(id) no. 74

(id) no. 75

(id) no. 76

(id) no. 77

(id) no. 78

(id) no. 79

(id) no. 80

(id) no. 81

(id) no. 82

(id) no. 83

(id) no. 84

(id) no. 85

(id) no. 86

(id) no. 87

(id) no. 88

(id) no. 89

(id) no. 90

(id) no. 91

(id) no. 92

(id) no. 93

(id) no. 94

(id) no. 95

(id) no. 96

(id) no. 97

(id) no. 98

(id) no. 99

(id) no. 100

(id) no. 101

(id) no. 102

(id) no. 103

(id) no. 104

(id) no. 105

(id) no. 106

(id) no. 107

(id) no. 108

(id) no. 109

(id) no. 110

(id) no. 111

(id) no. 112

(id) no. 113

(id) no. 114

(id) no. 115

(id) no. 116

(id) no. 117

(id) no. 118

(id) no. 119

(id) no. 120

(id) no. 121

(id) no. 122

(id) no. 123

(id) no. 124

(id) no. 125

(id) no. 126

(id) no. 127

(id) no. 128

(id) no. 129

(id) no. 130

(id) no. 131

(id) no. 132

(id) no. 133

(id) no. 134

(id) no. 135

(id) no. 136

(id) no. 137

(id) no. 138

(id) no. 139

(id) no. 140

(id) no. 141

(id) no. 142

(id) no. 143

(id) no. 144

(id) no. 145

(id) no. 146

(id) no. 147

(id) no. 148

(id) no. 149

(id) no. 150

(id) no. 151

(id) no. 152

(id) no. 153

(id) no. 154

(id) no. 155

(id) no. 156

(id) no. 157

(id) no. 158

(id) no. 159

(id) no. 160

(id) no. 161

(id) no. 162

(id) no. 163

(id) no. 164

(id) no. 165

(id) no. 166

(id) no. 167

(id) no. 168

(id) no. 169

(id) no. 170

(id) no. 171

(id) no. 172

(id) no. 173

(id) no. 174

(id) no. 175

(id) no. 176

(id) no. 177

(id) no. 178

(id) no. 179

(id) no. 180

(id) no. 181

(id) no. 182

(id) no. 183

(id) no. 184

(id) no. 185

(id) no. 186

(id) no. 187

(id) no. 188

(id) no. 189

(id) no. 190

(id) no. 191

(id) no. 192

(id) no. 193

(id) no. 194

(id) no. 195

(id) no. 196

(id) no. 197

(id) no. 198

(id) no. 199

(id) no. 200

(id) no. 201

(id) no. 202

(id) no. 203

(id) no. 204

(id) no. 205

(id) no. 206

(id) no. 207

(id) no. 208

(id) no. 209

(id) no. 210

(id) no. 211

(id) no. 212

(id) no. 213

(id) no. 214

(id) no. 215

(id) no. 216

(id) no. 217

(id) no. 218

(id) no. 219

(id) no. 220

(id) no. 221

(id) no. 222

(id) no. 223

(id) no. 224

(id) no. 225

(id) no. 226

(id) no. 227

(id) no. 228

(id) no. 229

(id) no. 230

(id) no. 231

(id) no. 232

(id) no. 233

(id) no. 234

(id) no. 235

(id) no. 236

(id) no. 237

(id) no. 238

(id) no. 239

(id) no. 240

(id) no. 241

(id) no. 242

(id) no. 243

(id) no. 244

(id) no. 245

(id) no. 246

(id) no. 247

(id) no. 248

(id) no. 249

(id) no. 250

(id) no. 251

(id) no. 253

S.No:25
P. Sathwik

Procedure T()

```
{
    if input symbol / ("*") then
        advance();
        F();
        T();
    }
}
```

$T \rightarrow *FT$

3 | 'T' + ← 'T'

'T' + ← 'T'

3 | 'T' + ← 'T'

Procedure F()

```
{
    if input symbol = 'id' then
        advance();
    else if input symbol = 'c', then
        advance();
        E();
    else if input symbol = ')'
        advance();
    else error();
}
```

$F \rightarrow id$

3 | 'T' + ← 'T'

$F \rightarrow (E)$

(error) - leading token to

: () T

: () T

: () T

: () T

: () T

: () T

② Construct the follow's grammar for operator procedure parser.

Sol:

	a	()	,	\$
a	>	>	>	>	>
(<	>	>	>	>
)	<	>	>	>	>
,	<	<	>	>	>
\$	<	<	<	<	

Step-1

$\$ (a, (a, a)) \$$

We insert procedure operator b/w symbol

$\$ <(<a>, <(<a>, <a>) >) > \$$

Step-2:

We can scan and parse string as

$\$ <(<a>, <(<a>, <a>) >) > \$$

$\$ <(s_1 <(<a>, <a>) >) > \$$

$\$ <(s_1 <(s_1 <a>) >) > \$$

$\$ <(s_1 <(s_1 s) >) \$$

$\$ <(s_1 <(l, s) >) > \$$

$\$ <(s_1 <(l) >) > \$$

$\$ <(s_1 s) > \$$

$\$ <(l, s) > \$$

$\$ <(l) > \$$

$\$ <s > \$$

$\$ \$$

③ Design the dependences graph for the following
expressions.

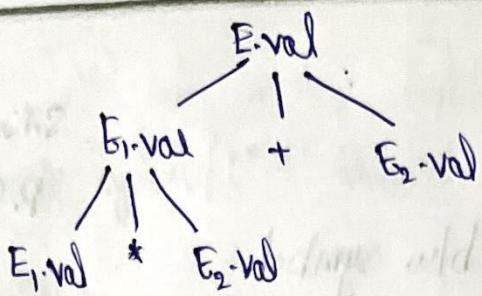
$$E \rightarrow E_1 + E_2$$

$$E \rightarrow E_1 * E_2$$

Production	Semantic Rule
$E \rightarrow E_1 + E_2$	$E.\text{val} \rightarrow E_1.\text{val} + E_2.\text{val}$
$E \rightarrow E_1 * E_2$	$E.\text{val} \rightarrow E_1.\text{val} * E_2.\text{val}$

S.No: 25

P. Sathwik



④ Design Run dependency graph for the following grammar.

$S \rightarrow T \text{ List}$

$T \rightarrow \text{int}$

$T \rightarrow \text{float}$

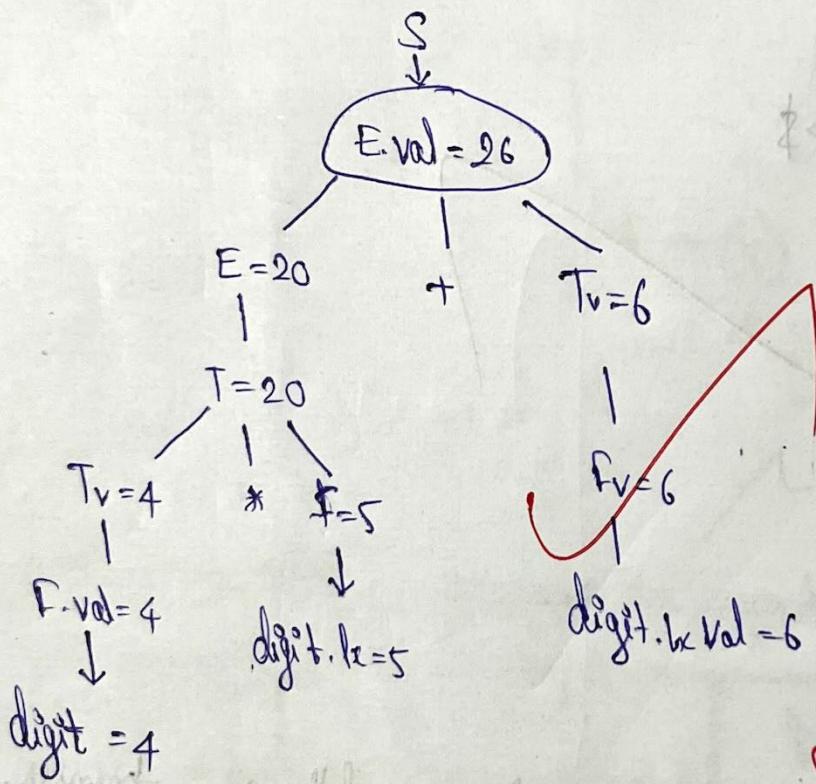
$T \rightarrow \text{class}$

$T \rightarrow \text{double}$

List \rightarrow list, id

List \rightarrow id

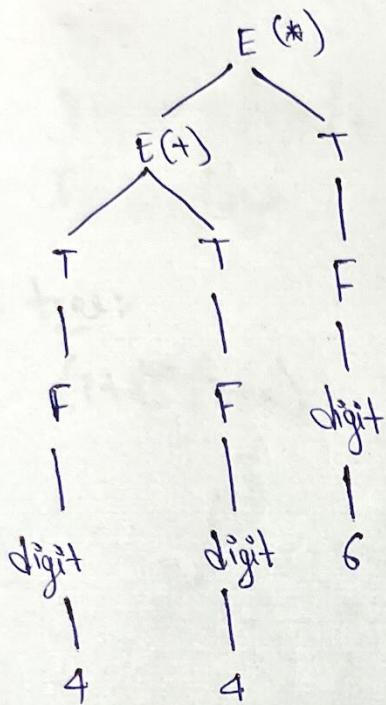
Design for above grammar of dependency graph.



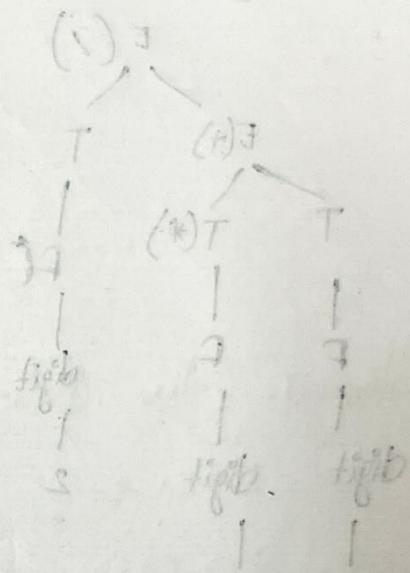
Analytical Day-08

① SDD for simple desk. Parse tree for expression
 $(6+4) * (4+3)$. flow. tipib = low + high S.No: 25

Parse Tree:



$\downarrow (E^* z.E + A)$



SDD:

$S \rightarrow E \cdot id$

$E \rightarrow E \cdot T \mid E \cdot -T \mid T$

$T \rightarrow T \cdot *F \mid T \cdot /F \mid T \cdot F$

$F \rightarrow (E) \mid digit$

$S \rightarrow E \quad \{ n \cdot val = E \cdot val \}$

$E \rightarrow E \cdot T \quad \{ E \cdot val = E \cdot val + T \cdot val \}$

$E \rightarrow E \cdot -T \quad \{ E \cdot val = E \cdot val - T \cdot val \}$

$E \rightarrow T \quad \{ E \cdot val = T \cdot val \}$

$T \rightarrow T \cdot *F \quad \{ T \cdot val = T \cdot val * F \cdot val \}$

$T \rightarrow T \cdot /F \quad \{ T \cdot val = T \cdot val / F \cdot val \}$

$T \rightarrow F \quad \{ T \cdot val = F \cdot val \}$

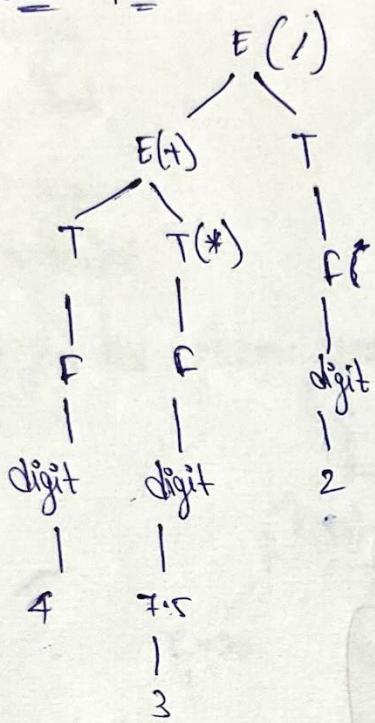
$f \rightarrow (E)$ if $F \cdot \text{val} = E \cdot \text{val}$ then define $S, N; 25$

$$F \rightarrow \text{digit} \quad \left\{ F.\text{val} = \text{digit}.\text{val}\right\}$$

(2) Parse Tree according to syntax directed.

$$(4 + 7.5 * 3) / 2$$

A. Parse Tree:



$$\text{Evaluate } (7.5 * 3) = 22.5$$

$$\text{Evaluate } (4 + 22 \cdot 5) = 26 \cdot 5$$

$$\text{Evaluate } (26.5 / 2) = 13.25$$

$$\text{output} = 13.25$$

③ SDD for operators + and * variable

$$x^*(3^*x + x+x)$$

S → E

$$E \rightarrow E + T$$

E → T

$$T \rightarrow T + c$$

16

Analytical Day-09

SDD for simple desk parse tree $(T+4)^* (3+6)n$.

$S \rightarrow F$ *for integer val ex- S.No: 25*

$E \rightarrow E+T \mid E$

$T \mid T$

$T \rightarrow T^* F \mid T \mid F \mid F$

$F \rightarrow (E) \mid \text{digit}$

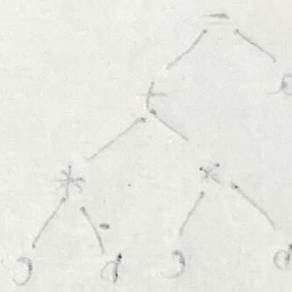
$2k+2^*d=0$

$2^*d + 2kd = 0$

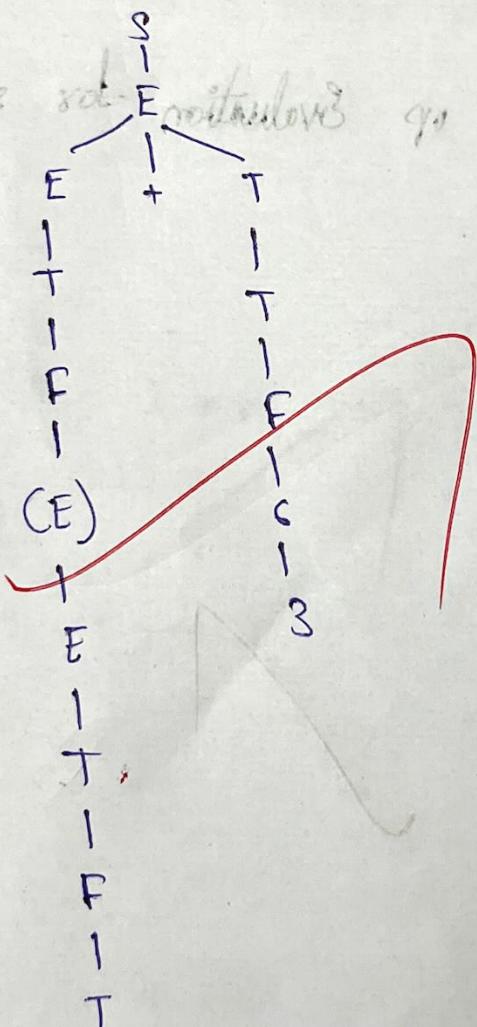
A

A) Parse tree:-

$(T+4)^* (3+6)$



• *decide which rule to apply at each level*



$+ \rightarrow 2^* 2$

$n \in \mathbb{N} \leq 2$

$T \rightarrow 3 \leq 3$

$T \rightarrow 3 \leq 3$

$T \leq 3$

$T^* T \leq T$

$T \mid T \leq T$

$T \leq T$

$3 \leq 3$

digit $\leftarrow 3$

$S \rightarrow E \{ n.\text{val} = E.\text{val} \}$

$E \rightarrow E_1 \{ E_1.\text{val} = E_1.\text{val} \} + T \{ E.\text{val} = E_1.\text{val} + T.\text{val} \} \mid T \{ t.\text{val} = T.\text{val} \}$

$T \rightarrow T_1 \{ T.\text{val} = T.\text{val} \} ^* F \{ T.\text{val} = T_1.\text{val} + F.\text{val} \} \mid F \{ T.\text{val} = F.\text{val} \}$

$F \rightarrow (E \{ E \cdot val = E \cdot val \}) \mid \text{digit } \{ F \cdot val = \text{digit} \cdot val \}$

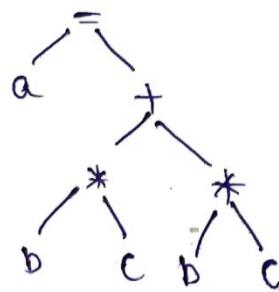
S.No: 25
P.O Pathankot

Q) Construct the syntax tree for expression $a = b * c + b * c$

$$a = b * c + b * c$$

A:

$$a = b * c + b * c$$



Q) Illustrate the bottom up Evaluation for subtraction

$$5 * 6 + 4$$

$$S \rightarrow EN$$

$$E \rightarrow E + T$$

$$E \rightarrow E - T$$

$$E \rightarrow T$$

$$T \rightarrow T * F$$

$$T \rightarrow F / F$$

$$T \rightarrow F$$

$$F \rightarrow E$$

$$E \rightarrow \text{digits}$$



Grammar:

Ques. 2) Given below is a grammar for decimal numbers.

$$S \rightarrow EN$$

Input : $5 * 6 + 4$

Step 1: Shift the digit in stack. S.No: 25

Stack: [5]

Input: * 6 + 4

Step 2: Shift the * in stack

Stack: [5, *]

Input: 6 + 4

Step 3: Shift the 6 number in stack

Stack: [5, *, f]

Input: + 4

Step 4: Redu E \rightarrow T

Stack: [E]

Input: + 4

Step 5: [E, +]

4

Step 6: F \rightarrow E \rightarrow digit

[E, +, f]

Step 7: Redu T \rightarrow F

[E +, f)

F²
P.K.8

P. Dathukk

F² twqL

lov.3 = dvi.2

lov.7 = dvi.13

lov.7 = dvi.17

lov.7 = dvi.37

Z = ebw.7, Z = lov.7
F = ebw.7, F = lov.7

T \rightarrow F (f \rightarrow E \rightarrow digit)
F = ebw.7, F = lov.7

Z = ebw.7, Z = lov.7

Z = ebw.7, Z = lov.7

Z = ebw.7, Z = lov.7

2 \times 2 = 46

+ + 8 twqL

lov.3 = dvi.2

lov.7 = dvi.13

lov.7 = dvi.17

lov.7 = dvi.37

Z = ebw.7, Z = lov.7

D = dvi.7, D = lov.7

8 - dvi.7, 8 = lov.7

S = ebw.7, S = lov.7

* 2 : 9/0

(4) SDD scheme that translates

5 * 7

8 + 4

S.No: 25

P. Sathwik

A

Grammar:-

1. Input 5 * 7

S.inh = E.val

E1.inh = T.val

T1.inh = F.val

T2.inh = F.val

F1.val = 5, F1.code = 5

F1.val = 7, F2.code = 7

T2.val = 7, T2.code = 7

T1.val = 5, T1.code = 5

E1.val = 5, E1.code = 5

E.val = 5, E.code = 5

O/P = 57*

2. Input 8 + 4

S.inh = E.val

E1.inh = T.val

T1.inh = F.val

T2.inh = F2.val

F1.val = 8, F1.code = 8

F2.val = 4, F2.code = 4

T2.val = 4, T2.code = 8

E1.val = 8,

E.val = 8, E.code = 8

O/P : 84*

Analytical Day-10

① Quadruples for expression $(a+b) + (c+d) - (a+b)(c+d)$

Let,

$$t_1 = a+b$$

$$t_2 = -t_1$$

$$t_3 = c+d$$

$$t_4 = t_2 * t_3$$

$$t_5 = t_1 + c$$

$$t_6 = t_4 - t_5$$

S.No: 25

P. Sathwik

Quadruples:

	Operator	Arg ₁	Arg ₂	Result
0	+	a	b	t ₁
1	-	t ₁		t ₂
2	+	c	d	t ₃
3	+	t ₂	t ₃	t ₄
4	+	t ₁	c	t ₅
5	-	t ₄	t ₅	t ₆

② Construct the address code following executed if A < B
and C < D then t=1 else t=0

If A < B and C < D

$$t = 1$$

else

$$t = 0$$

These address code

(1) if ($A < B$) goto 3

S.No: 25

(2) goto (4)

P. Sathwik

(3) if ($C < 0$) goto 6

dr0 = 1

(4) $t = 0$

dr1 = dr0

(5) goto (7)

dr2 = dr1

(6) $t = 1$

dr3 = dr2

(3) Generator the address code

~~do~~ C = 0

do
for

if ($a < b$) then

x++;

else

x--;

c++;

while ($c < 5$)

Ans There address code

① $c = 0$

7 $T_2 = x - 1$

② if ($a < b$) goto (4)

8 $x = 12$

③ goto (4)

9 $T_8 = c + 1$

④ $T_6 (x + 5)$

10 $c = T_3$

⑤ $x = T_1$

11 if ($c < 5$)

⑥ goto (9)

12 Goto (2)

④ Generate three address code

int a[40], b[10], i, d_p=0;

S.No:25

for (i=0; i<10; i++)

P. Sathwik

{
d_p += a[i] + b[i];

}

① i=0

② t₁ = i < 10

③ if not t₁ goto 9

4. t₂ = i+4

5. t₃ = a[t₂]

6. t₄ = i+4

7. t₅ = b[t₄]

8. t₆ = t₃ + t₅

9. t₇ = d_p + t₆

10. d_p = t₇

11. i = i+1

12. goto 2.

14/12/23