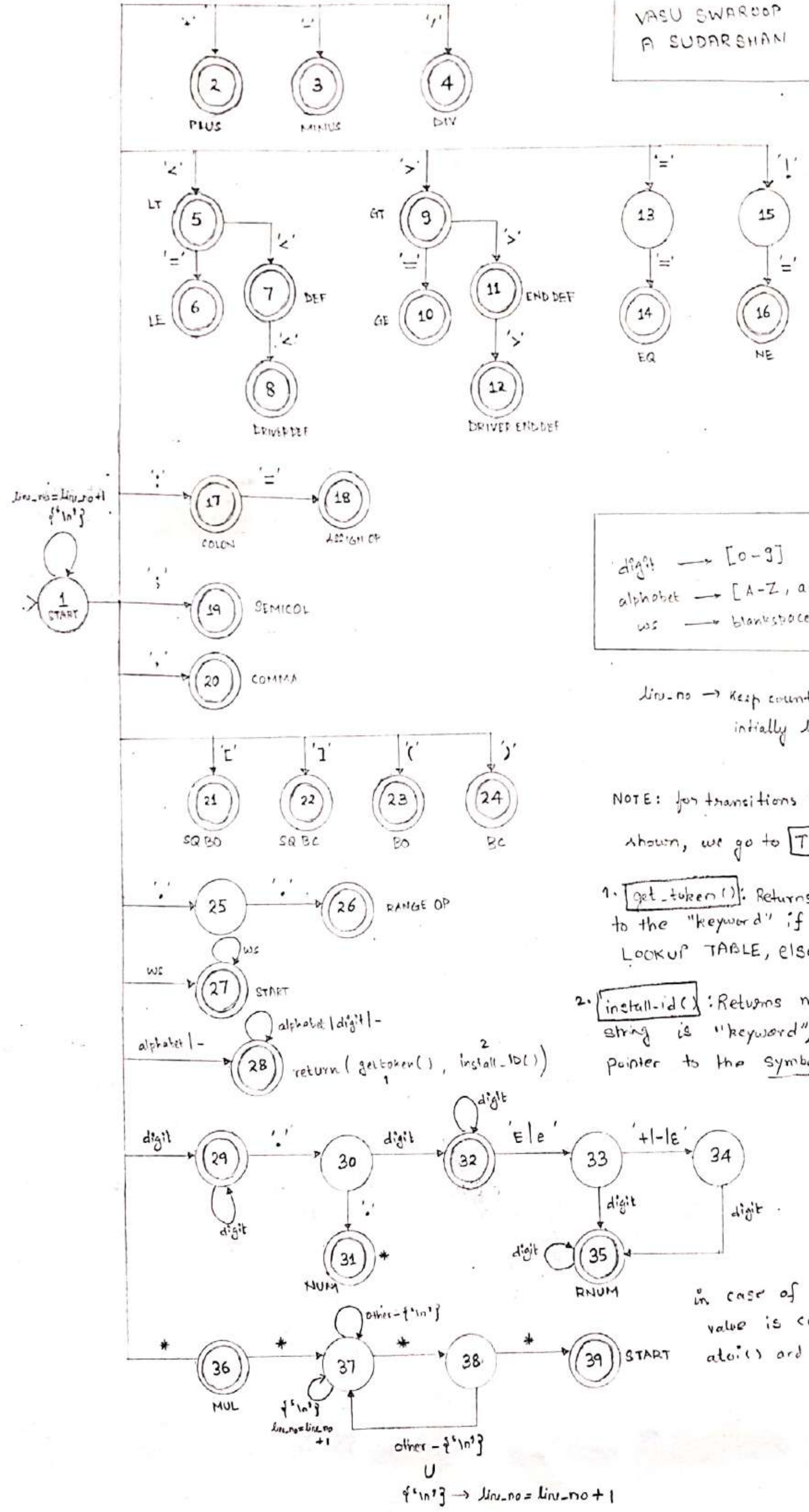
Name	[illegible]
RAJAN SAHU	0572
YASH GOYAL	0638
AYUSH AGARWAL	0652
VASU SWAROOP	0656
A SUDARSHAN	0744



digit → [0-9]
alphabet → [A-Z, a-z]
ws → blankspace | tab | '\r'

line_no → Keep count of line number,
initially line_no = 1.

NOTE: for transitions that are not shown, we go to TRAP state..

1. get_token(): Returns token correspond to the "keyword" if present in the LOOKUP TABLE, else returns ID

2. install_id(): Returns null if matched string is "keyword", else return pointer to the symbol_table entry.

in case of NUM and RNUM, value is computed using atoi() and atof() respectively

P.T.O

Name	ID (2019B4A7____)
1. Rajan Sahu	(0572)
2. Ayush Agarwal	(0652)
3. Yash Goyal	(0638)
4. Vasu Swaroop	(0656)
5. A Sudarshan	(0744)

LOOKUP TABLE

PATTERN	TOKEN
integer	INTEGER
real	REAL
boolean	BOOLEAN
of	OF
array	ARRAY
start	START
end	END
declare	DECLARE
module	MODULE
driver	DRIVER
program	PROGRAM
get_value	GET_VALUE
print	PRINT
use	USE
with	WITH
parametras	PARAMETERS
takes	TAKES
input	INPUT
returns	RETURNS
for	FOR
in	IN
switch	SWITCH
case	CASE
break	BREAK
default	DEFAULT
while	WHILE
AND	AND
and	AND
OR	OR
or	OR
true	TRUE
false	FALSE

1. Accept State without retraction ◎ → lookahead pointer must be kept on hold for one iteration of while loop containing switch cases.

2. Accept State with retraction ◎* → lookahead pointer must be kept on hold for one iteration of while loop containing switch cases and lookaheak pointer must be decremented once.

GROUP-20
DATE - 19/2/23

MEMBERS	
RAJAN-	2019B4A70572P
YASH-	2019B4A70638P
AYUSH-	2019B4A70652P
VASU-	2019B4A70656P
SUDARSHAN-	2019B4A70744P

GRAMMAR

<start> → <program> $

1. <program> → <moduleDeclarat^{ns}> <otherModules> <driverModule> <otherModules>
2. <moduleDeclarat^{ns}> → <moduleDeclarat^{n}> <moduleDeclarat^{ns}> | ϵ
3. <moduleDeclarat^{n}> → DECLARE MODULE ID SEMICOL
4. <otherModules> → <module> <otherModules> | ϵ
5. <driverModule> → DRIVERDEF DRIVER PROGRAM DRIVERENDDEF <moduleDef>
6. <module> → DEF MODULE ID ENDDEF TAKES INPUT SQBO <input_plist> SQBC SEMICOL <ret> <moduleDef>
7. <ret> → RETURNS SQBO <output_plist> SQBC SEMICOL | ϵ
8. <input_plist> → ID COLON <dataType> <ip>
9. <ip> → COMMA ID COLON <dataType> <ip> | ϵ
10. <output_plist> → ID COLON <type> <op>
11. <op> → COMMA ID COLON <type> <op> | ϵ
12. <dataType> → INTEGER
13. <dataType> → REAL
14. <dataType> → BOOLEAN
15. <dataType> → ARRAY SQBO <range> SQBC OF <type>
16. <type> → INTEGER
17. <type> → REAL
18. <type> → BOOLEAN
19. <moduleDef> → START <statements> END
20. <statements> → <statement> <statements> | ϵ
21. <statement> → <ioStmt>
22. <statement> → <simpleStmt>
23. <statement> → <declareStmt>
24. <statement> → <conditionalStmt>
25. <statement> → <iterativeStmt>
26. <ioStmt> → GET_VALUE BO ID BC SEMICOL
27. <ioStmt> → PRINT BO <var"> BC SEMICOL
28. <var"> → ID <whichID>
29. <var"> → NUM
30. <var"> → RNUM
31. <var"> → TRUE
32. <var"> → FALSE

33.<whichID> → SQBO <which'> SQBC | ϵ
34.<which'> → <type'>
35.<which'> → <arithmetic Expression'>
36.<type'> → NUM
37.<type'> → ID
38.<simplestmt> → <assigoment Stmt>
39.<simple Stmt> → <module Reuse Stmt>
40.<assigment Stmt> → ID <which Stmt>
41.<which Stmt> → <Lvalue IDStmt>
42.<which Stmt> → <lvalue ARR Stmt>
43.<LValue IDStmt> → ASSIGNOP <expression> SEMICOL
44.<lValue ARRStmt> → SQBO <index> SQBC ASSIGNOP <expression> SEMICOL
45.<index> → <arithmetic Expression'>
46.<moduleReuseStmt> → <optional> USE MODULE ID WITH PARAMETERS <idList> SEMICOL
47.<optional> → SQBO <idList> SQBC ASSIGNOP | ϵ
48.<idList> → ID <idList'>
49.<idList'> → COMMA ID <idList'> | ϵ
50.<expression> → <arithmetic Expression>
51.<expression> → <boolean Expression>
52.<arithmeticExpression> → <term> <ae>
53.<arithmetic Expression> → PLUS <newterm>
54.<arithentic Expression> → MINUS <newterm>
<ae> → <op'> <term> <ae> | ϵ
56.<term> → <factor> <te>
57.<te> → <op''> <factor> <te> | ϵ
58.<factor> → BO <arithmetic Expression> BC
59.<factor> → <var>
60.<var> → ID <whichID>
61.<var> → NUM
62.<var> → RNUM
63.<op'> → PLUS
64.<op'> → MINUS
65.<op''> → MUL
66.<op''> → DIV
67.<arithmetic Expression'> → <term'> <ae'>
68.<arithmetic Expression'> → PLUS <newterm'>
69.<arithmetic Expression'> → MINUS <newterm'>

```
<ae'> → <op'><term'><ae'> | ϵ
<term'> → <factor'><te'>
72. <te'> → <op''> <factor'> <te'> | ϵ
73. <factor'> → BO <arithmeticExpression'> BC
74. <factor'> → <var'''>
75. <var'''> → ID
76. <var'''> → NUM
77. <var'''> → RNUM
78. <newterm> → ID
79. <newterm> → NUM
80. <newterm> → RNUM
81. <newterm> → Bo <arithmeticExpression> BC
82. <newterm'> → ID
83. <newterm'> → NUM
84. <newterm'> → RNUM
85. <newterm'> → BO <arithmetic Expression'> BC
86. <booleanExpr> → <arithmeticExpression> <booleanExpr'>
87. <booleanExpr'> → <relationalOp> <arithmeticExpression> <bE> | ϵ
88. <bE> → <logical Op> <arithmetic Expression> <relational Op> <arithmeticExpression> <bE> | ϵ

89. <booleanExpr> → BO <boolean Expr.> BC <be'>

90. <be'> → <logicalOp> BO <boolean Expr> BC <be'> | ϵ
91. <be'> → TRUE
92. <be'> → FALSE
93. <be'> → ID

94. <logicalOp> → AND | OR

95. <relational Op> → LT
96. <relational Op> → LE
97. <relational Op> → GT
98. <relational Op> → GE
99. <relational op> → EQ
100. <relational op> → NE
101. <declareStmt> → DECLARE <idList> COLON <dataType> SEMICOL
102. <conditional Stmt> → SWITCH BO ID BC START <caseStmt> <default> END
103. <case Stmt> → CASE <int.bool>
```

104. <int_bool> → <int>

105. <int_bool> → <bool>

106. <int> → NUM COLON <statements> BREAK SEMICOL <r_int>

107. <r_int> → CASE <int>

108. <r_int> → DEFAULT

109. <bool> → <value> COLON <statements> BREAK SEMICOL <r_bool>

110. <r_bool> → CASE <bool> | ϵ

111. <value> → TRUE

112. <value> → FALSE

113. <default> → DEFAULT COLON <statements> BREAK SEMICOL | ϵ

114. <iterativeStmt> → FOR BO ID IN <range> BC START <statements> END.

115. <iterativeStmt> → WHILE BO <booleanExpr> BC START <statements> END.

116. <range> → NUM RANGEOP NUM

117. <range'> → <type'> RANGEOP <type'>

Assumptions.

1. i := +3+2 and similar statements are not allowed.
2. ++a , --a are not allowed.
3. −(−(a+b)) and similar statements are allowed.
4. A[+(2+5)] indexing is allowed

FIRST SETS

FIRST (<start>) = {DECLARE, DEF, DRIVERDEF}
FIRST(<program>) = {DECLARE, DEF, DRIVERDEF}
FIRST (<moduleDeclerations>) = {DECLARE, ε}
FIRST(<module Declaration>) = {DECLARE}
FIRST (<other Moducl>) = {DEF, ε}
FIRST (<driverModule>) = {DRIVERDEF}
FIRST (<module>) = {DEF}
FIRST (<ret>) = {RETURNS, ε}
FIRST(<input_plist>) = {ID}
FIRST (<ip>) = {COMMA, ε}
FIRST (<output_plist>) = {ID}
FIRST(<op>) = {COMMA, ε}
FIRST (<datatype>) = {INTEGER, REAL, BOOLEAN, ARRAY}
FIRST(<type>) = {INTEGER, REAL, BOOLEAN}
FIRST (<moduleDef>) = {START}
FIRST (<statements>) = {GET_VALUE, PRINT, ID, SQBO, USE, DECLARE,
SWITCH, FOR, WHILE, ε}
FIRST (<statement>) = {GET_VALUE, PRINT, ID, SQBO, USE, DECLARE, SWITCH, FOR,
WHILE}
FIRST (<ioStmt>) = {GET_VALUE, PRINT}
FIRST (<var''>) = {ID, NUM, RNUM, TRUE, FALSE}
FIRST(<whichID>) = {SQBO, ε}
FIRST(<which'>) = {NUM, ID, BO, RNUM, PLUS, MINUS}
FIRST (<type'>) = {NUM, ID}
FIRST(<simpleStmt>) = {ID, SQBO, USE}
FIRST(<assignmentStmt>) = {ID}
FIRST (<whichStmt>) = {ASSIGNOP, SQBO}
FIRST(<lValueIdStmt>) = {ASSIGNOP}
FIRST(<lValueArrStmt>) = {SQBC}
FIRST(<index>) = {NUM, ID, BO, RNUM, PLUS, MINUS}
FIRST(<moduleReuseStmt>) = {SQBO, USE}

FIRST (<optional>) = {SQBO, ∈}

FIRST (<idList>) = {ID}

FIRST (<idList'>) = {COMMA, ∈}

FIRST(<expression>) = {BO, ID, NUM RNUM, PLUS, MINUS, LT, LE, GT, GE, EQ, NE, ∈, }

FIRST (<arithmeticExpression>) = {BO, ID, NUM, RNUM, PLUS, MINUS}

FIRST (<ae>) = {PLUS, MINUS, ∈}

FIRST (<term>) = {BO, ID, NUM, RNUM}

FIRST(<te>) = {MUL, DIV, ∈}

FIRST (<factor>) = {BO, ID, NUM, RNUM}

FIRST (<var>) = {ID, NUM, RNUM}

FIRST (<op'>) = {PLUS, MINUS}

FIRST(<op''>) = {MUL, DIV}

FIRST (<arithmeticExpression'>) = {BO, ID, NUM, RNUM, PLUS, MINUS}

FIRST (<ae'>) = {PLUS, MINUS, ∈}

FIRST(<term'>) = {BO, ID, NUM, RNUM}

FIRST(<te'>) = {MUL, DIV, ∈}

FIRST(<factor'>) = {ID, NUM, RNUM, BO}

FIRST (<var'''>) = {ID, NUM, RNUM}

FIRST(<newTerm>) = {ID, NUM, RNUM, BO}

FIRST (<newTerm'>) = {ID, NUM, RNUM, BO}

FIRST(< booleanExpr>) = {LT, LE, GT, GE, EQ, NE, ∈}

FIRST(<booleanExpr'>) = {LT, LE, GT, GE, EQ, NE, ∈}

FIRST (< bE >) = {AND, OR, ∈}

FIRST (< bE'>) = {AND, OR, ∈, TRUE, FALSE, ID}

FIRST (<logical Op>) = {AND, OR}

FIRST (<relationalOp>) = {LT, LE, GT, GE, EQ, NE}

FIRST(< declare Stmt>) = {DECLARE}

FIRST(< conditional Stmt >) = {SWITCH}

FIRST (<caseStmt> = {CASE}

FIRST(<int-bool>) = {NUM, TRUE, FALSE}

FIRST (< int >) = {NUM}

FIRST (<r_int >) = {CASE, DEFAULT}

FIRST(<bool>) = {TRUE, FALSE}

FIRST(<rbool>) = {CASE, ε}

FIRST(<value>) = {TRUE, FALSE}

FIRST(<default>) = {DEFAULT, ε}

FIRST(<iterativeStmt>) = {FOR, WHILE}

FIRST(<range>) = {NUM}

FIRST(<range'>) = {NUM, ID}

FOLLOW SETS

1. FOLLOW (<moduleDeclerations>) = {DEF, DRIVERDEF}
2. FOLLOW (<otherModules>) = {DRIVERDEF, $}
3. FOLLOW (<ret>) = {START}
4. FOLLOW (<ip>) = {SQBC}
5. FOLLOW (<program>) = {$}
6. FOLLOW (<input_plist>) = {SQBC}
7. FOLLOW (<op>) = {SQBC}
8. FOLLOW (<output_plist>) = {SQBC}
9. FOLLOW (<statements>) = {END, BREAK}
10. FOLLOW (<whichID>) = {MUL, DIV, PLUS, MINUS, SEMICOL, BC, LT, LE, GT, GE,
EQ, NE, AND, OR}
11. FOLLOW (<var'>) = {BC}
12. FOLLOW (<optional>) = {USE}
13. FOLLOW (<idList'>) = {COLON, SEMICOL, SQBC, COMMA}
14 FOLLOW (<idList>) = {COLON, SEMICOL, SQBC, COMMA}.
15* FOLLOW (<arithmeticExpression>) = {SEMICOL, BC, LT, LE, GT, GE, EQ
NE, AND, OR}
16. FOLLOW (<ae>) = {SEMICOL, BC, LT, LE, GT, GE, EQ, NE, AND
, OR}
17. FOLLOW (<be'>) = {BC, SEMICOL}
18. FOLLOW (<booleanExpr>) = {BC, SEMICOL}
19. FOLLOW (<te>) = {PLUS, MINUS, SEMICOL, BC, LT, LE, GT, GE, EQ
NE, AND, OR}
20. FOLLOW (<term>) = {PLUS, MINUS, SEMICOL, BC, LT, LE, GT, GE
EQ, NE, AND, OR}
21. FOLLOW (<var>) = {MUL, DIV, PLUS, MINUS, SEMICOL, BC, LT, LE, GT, GE,
EQ, NE, AND, OR}
22. FOLLOW (<factor>) = {MUL, DIV, PLUS, MINUS, SEMICOL, BC, LT, LE, GT, GE, EQ
NE, AND, OR}

23. FOLLOW(<ae'>) = {BC, SQBC}

24. FOLLOW(<arithmetic Expression'>) = {BC, $QBC}

25. FOLLOW(<which'>) = {SQBC}

26. FOLLOW (<index>) = {SQBC}

27. FOLLOW (<te'>) = {PLUS, MINUS, SQBC, BC}

28. FOLLOW (<term'>) = {PLUS, MINUS, SQBC, BC}

29. FOLLOW (<booleanExpr'>) = {BC, SEMICOL}

30. FOLLOW (<be>) = {BC, SEMICOL}

31. FOLLOW (<r_bool>) = {DEFAULT, END}

32. FOLLOW (<ind_bool>) = {DEFAULT, END}

33. FOLLOW (<caseStmt>) = {DEFAULT, END}

34. FOLLOW (<bool>) = {DEFAULT, END}

35. FOLLOW (<default>) = {COLON}