Name - Anoop kumar Sharma
CSE - 18/018

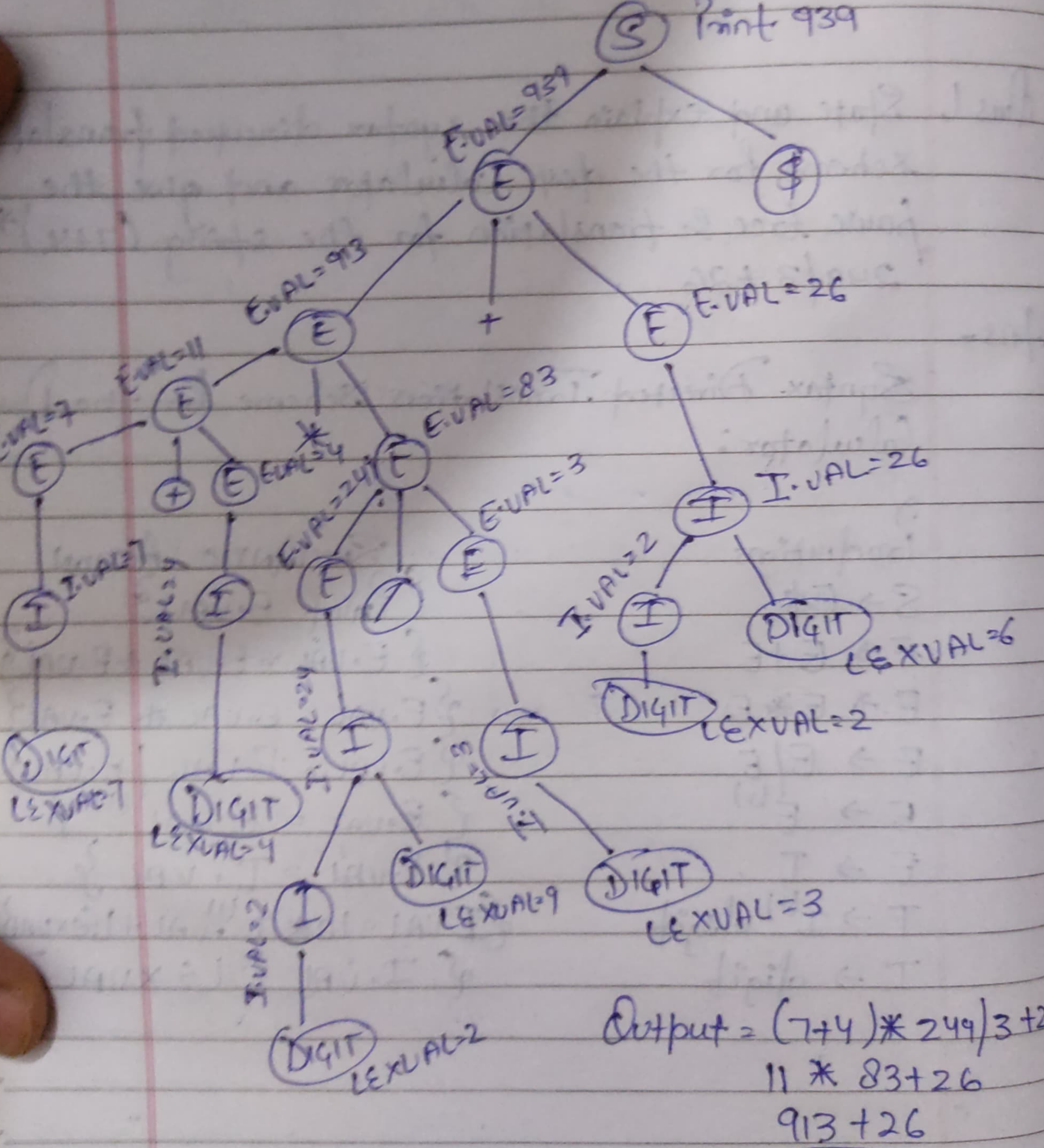
Ques 1. State and explain the syntax directed translation scheme for the desk Calculator and give the pause tree & translation for the string (7+4)* 249/3+26

Ans -

Syntax Directed Translation Scheme for the Desk Calculator:

Productions	SEMANTIC ACTION
S → E$	{ Paint E.VAL }
E → E+E	{ E.VAL = E.VAL + E.VAL }
E → E*E	{ E.VAL = E.VAL * E.VAL }
E → E/E	{ E.VAL = E.VAL / E.VAL }
E → $E^{(1)}$	{ E.VAL = $E.VAL^{(1)}$ }
E → I	{ E.VAL = I.VAL }
I → $I^{(1)}$ digit	{ I.VAL = 10 * $I^{(1)}$.VAL + LEXVAL }
I → digit	{ I.VAL = LEXVAL }

→ Parse Tree with Translation for the Input (7+4) * 249/3 +26



Output = (7+4)*249/3+26
11 * 83+26
913+26
$\boxed{939}$ Ans.

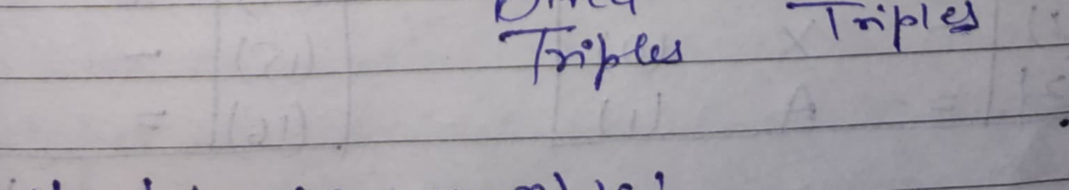
Ques 2. What is intermediate Code Representation? Explain Quadruple, Triples and Indirect Triples with the help of an example.

Ans → Intermediate Code Generation:

It recives "Annoted Syntax Tree" as an input from Analyser phase and i.e. Semantic Analyzer phase and converted the input into a linear representation such as Three-Address Code, Postfin Notation etc.

Three-Address Code

- It is one of the many ways to represte the Intermediate Code.
- It uses at most 3 location of address to Calulated the expression.



Lets take an example!

A = X - Y + Z;

3- Address Code of the given example

$T_1 = Y + Z$

$T_2 = X - T_1$

$A = T_2$

1) Quadruple

Here, each intruction is divided into 4 fields i.e. Operator ARG 1, ARG 2 and RESULT

OP	ARG(1)	ARG(2)	RESULT
+	Y	Z	T_1
-	X	T_1	T_2
=	T_2		A

2) Triple

Each intruction has 3 fields i.e. Operator ARG 1 and ARG 2

(i) Direct Triples

S.no	OP	ARG1	ARG2
(0)	+	Y	Z
(1)	-	X	(0)
(2)	=	A	(1)

(ii) Indirect Triples

	OP	ARG1	ARG2
(14)	+	Y	Z
(15)	-	X	(14)
(16)	=	A	(15)

STATEMENT

(0)	(4)
(1)	(15)
(2)	(16)

Ques 3. What is the use of Symbol Table? Explain any 2 data structure associated with Symbol Table

Ans # Symbol Table

Symbol Table is an important Data Structure Created and maintained by Compilers in Order to store information about the Occurrance of various entitles such as variable names (lexemes | Symbols) function names, Objects, Classes etc.

→ Uses of Symbol Table:

1. It stores the names of all entiles in a structure from at one place.
2. It's used to verify if a variable has been declared.
3. It also determined the scope of a name.
4. Helpful in implementing type checking, by verifying assignment and expression in two source code are semantically Concert.

Data Structure used for Implementing symbol Table are:

1) Hash Tables

- Two table: Hash Table and Sumbol Table, are maintained here.
- A hash table is an array within index range: 0 to Table.-size -1
- To search for a name, we use hash functions, that will result in any integer between 0 to table size -1.
- Quick search is possible here as insection and lookup can be made very fast - O(1).

2) Binary Seach Tree (BST)

- Here, we add two link fields i.e. left and right child to the parent.
- All names are created as child of root node that always follow the property of BST.
- It can grow dynamically.
- Insertion and Lookup are $O(\log_2 n)$ on average

Ques 4 Write a short note on

(a) Loop Optimization

(b) Peephole Optimization

Ans (a) Loop Optimization

- It is the process of increasing execution speen and reducing the Overheads associated with loops.
- It is a machine Independent Optimization used to improve cache performance.

Loop Optimization Techniques!

1) Frequency Reduction (Code Motion)
2) Loop Unrolling
3) Loop Jamming
4) Strength Reduction
5) Induction-Variable Eliminaction
6) Dead code Elimination.

b) Peephole Optimization

- It is a machine dependent optimization performed on small part of the code.
- It is applied after the generation of Target code in a repeated way.
- It basically replaces a part of code in a repeated way.
- It basically replaces a part of code with shorted and faster code without changing o/p.

Peephole Optimization Techniques

1. Redundant load and Store Elimination
2. Constant folding
3. Strength Reduction
4. Unreachable Code.
5. Algebric Simplication

Ques 6 Convert the following statement into the quadraple Triple and Indirect Triple representation:

$$A = - B * (C + D)$$

Ans ~~Quad~~ Qudruple Representation -.

OP	ARG1	ARG2	Result	3-Address Code
Uniminus	B	—	T_1	$T_1 = -B$
+	C	D	T_2	$T_2 = C+D$
*	T_1	T_2	T_3	$T_3 = T_1 * T_2$
=	T_3		A	$A = T_3$

Triple Representation

	OP	ARG1	ARG2
(0)	Uniminus	B	—
(1)	+	C	D
(2)	*	(0)	(1)
(3)	=	A	(2)

Indirect Triple Representation

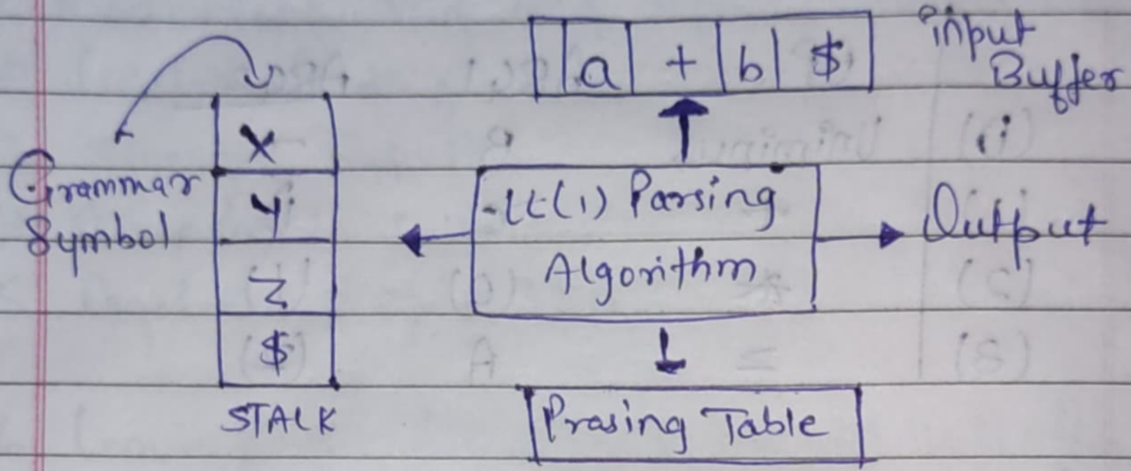
	STATEMENT		OP	ARG1	ARG2
(0)	(14)	(14)	-	B	-
(1)	(15)	(15)	+	C	D
(2)	(16)	(16)	*	(14)	(15)
(3)	(17)	(17)	=	A	(16)

Ques 7. Describe the following:

(a) Predective Pasing

- It is a Top-Down Parser with no backtracking or backup.
- It Constructs the pause tree from the top and the input is read from left to right.
- It has the Capability to predict which production is to be used to replace the input string.

Process - flow Diagram



(b) Olerator Precedence Parsing

- It is a Bottom - UP or shift - Reduce Parser that reads and understands an Operator Procedure Grammer
- It is established between the terminals of the grammer
- Any Grammer G is called an "operator Precedence Grammer" if!

→ No RHS of any production rule has E

→ No two non terminals are adjacent

Ques 8 Define SLR. write an Algorithm for the Construction of SLR Parsing Table.

Ans SLR parser:

→ A simple LR or SLR parser is a type of LR parser with small parse tables and a relatively

Simple parser generator algorithm. :

→ To Construct SLR(1) Parsing table, we use Canonical Collection of LR(0) item.

Algorithm to Construct SLR pars

Input: C { The Canonical Collection of items for an argumented grammer, G }

Output: If possible, an LR passing table Consisting of a parsing action function "ACTION" and a "GOTO" function.

METHOD:

```
1. Procedure CLOSURE (I):
   being
      repeat
            for each item $A \to \alpha \cdot B\beta$ in I and
            each production $B \to V$ in grammer G
            Such that $B \to V$ is not in I
               do add $B \to V$ to I ;
         Until no more items Can be added in I
            return I ;
   end
```

2. GOTO :

The function GOTO (I, x) where I is a set of items and x is a grammer symbol.

GOTO (I, x) is defined to be the cloure of two set of all item $[A \rightarrow \alpha x . \beta]$ Such that $[A \rightarrow \alpha x . \beta]$ is in I.

3. Procedure ITEMS (G):

```
begin:
    C := {CLOSURE ({S' -> S})};
  repeat
    for each set of item I in C and each
    grammer symbol X such that GOTO (I, X)
    is not empty and is not in C
       do add GOTO (I . X) to C
  Until no more sets of items can be added to C.
    end.
```