Model test

Imperial College of Engineering

Part-3 ....Semester: Odd Exam-2022

Student Name: Teresa Jency Bala

Student ID: 193852011 3

Subject Name: Compiler Design

Subject Code: CSE 3141 Session: 18-19

Department: Computer Science & Engineering.

48

Answer to the Q no-1(a).

Compiler:- A compiler is a computational component which can transform a high level language to a low level language. It takes the whole source code and ~~com~~ converts to a ~~lower~~ lower level language at once. Also an important ~~tak~~ task is to check for all the errors and displays the errors after the end of compilation

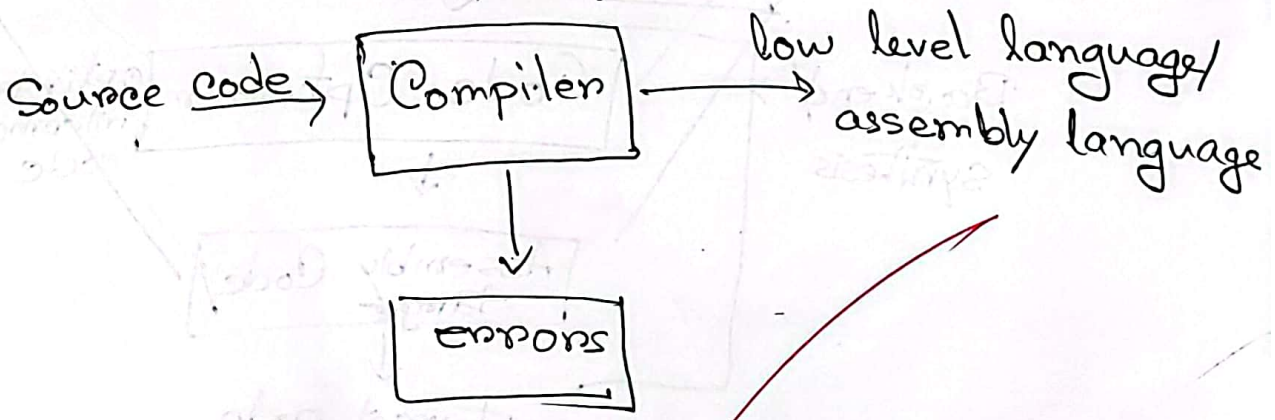
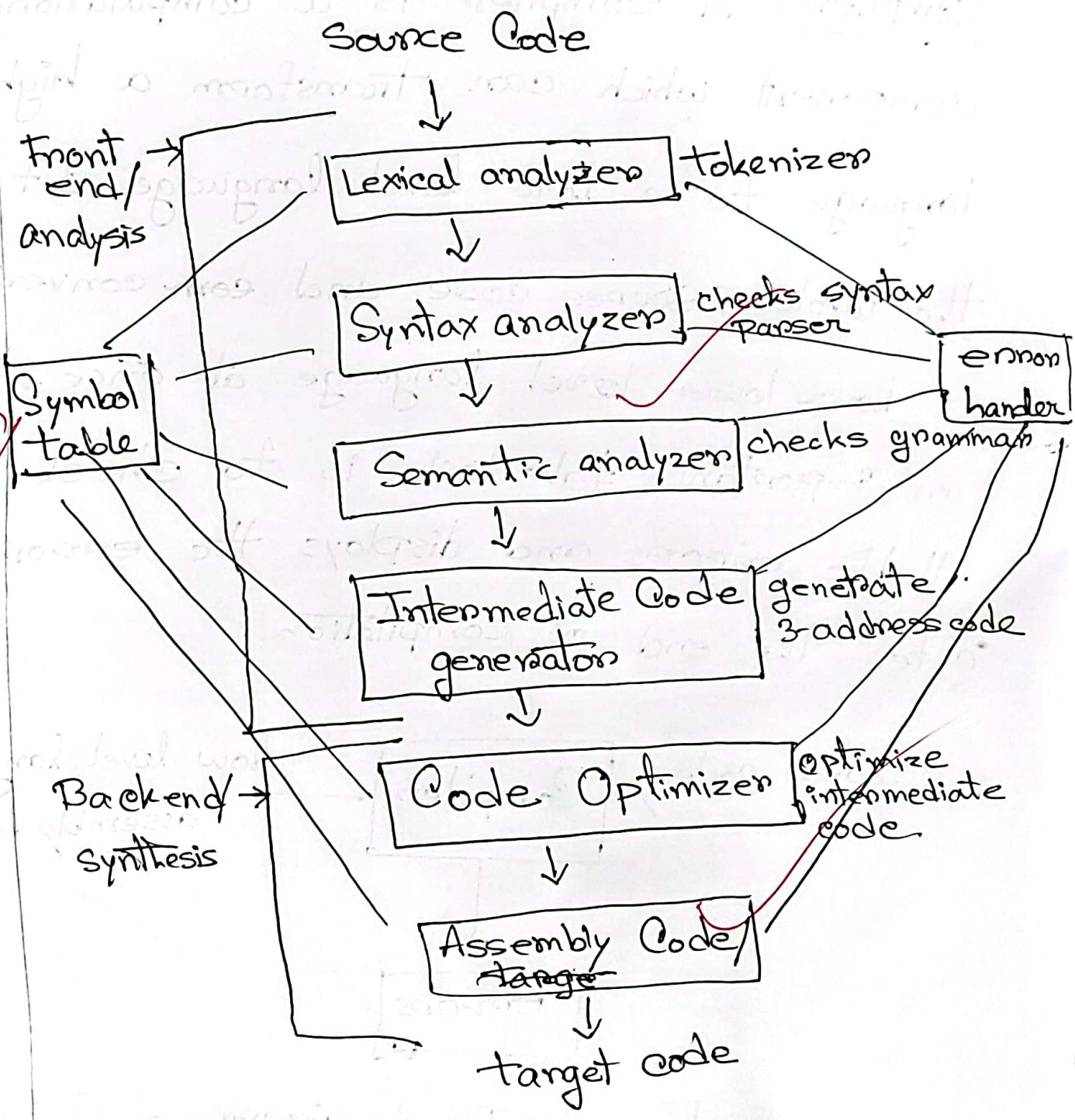


Figure: Block diagram of Compiler

Answer to the Qno-1(b)

There are mainly 6 phases of compiler



Answer to the Qno-1(c).

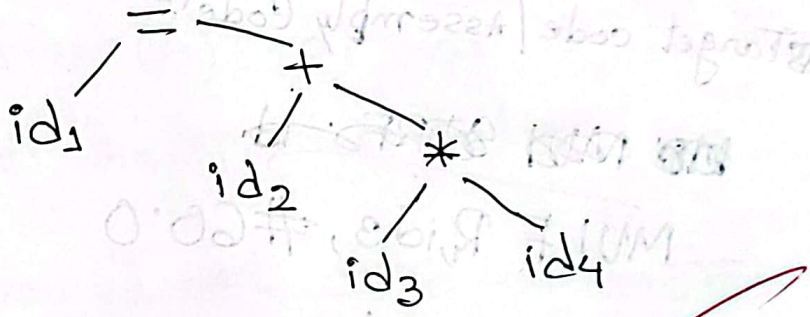
Given, value = initial + rate * 60

Laxical Analyzer:-

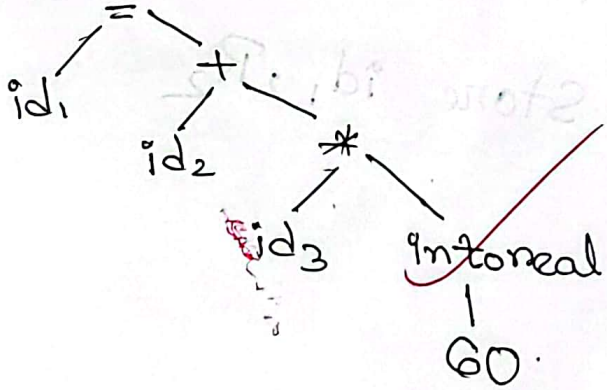
$id_1 = id_2 + id_3 * id_4$

5.25

Syntax Analyzer:-



Semantic Analyzer:-



Intermediate Code generator:-

$temp_1 := inttoreal(60)$.

$temp_2 := id_3 * temp_1$

$temp_3 = id_2 + temp_2$

$id_1 = temp_3$.

Code optimizer:-

$temp_1 = id_3 * 60.0$

$id_1 = id_2 + temp_1$.

Target code / Assembly Code:-

MULF R_1, id_3, #60.0

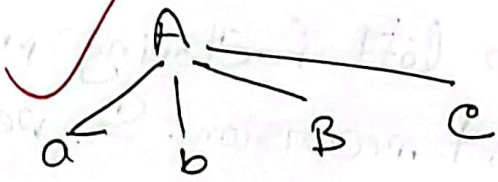
ADD R_2, R_1, id_2

Store id_1, R_2

Answer to the Qno- 3 (a).

The First() set of an grammar production is the set of all the terminals which are found in the starting position of the production line e.g.

$A \rightarrow abBc$

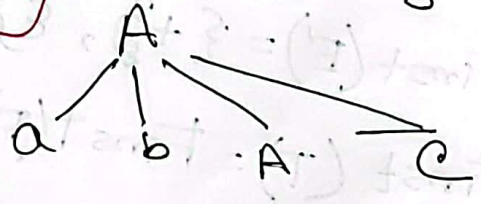


3.75

Heren First(A) = {a}, as it is the starting terminal of the production line

The follow() set of a production line is the non-terminal on the Right hand side of the production line which is immidiate next the Non-terminal which is in the follow(A). Eg.

$A \rightarrow abAc$



Follow(A) = {c}

here small c is terminal which follow A

Given, $E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

As the first check of the grammar is whether there is left-recursion or left factoring present. Here is left recursion. So, remove it

$A \to A\alpha \mid \beta$

$A \to \beta A'$

$A' \to \alpha A' \mid \varepsilon$

$E \to T E'$

$E' \to +TE' \mid \varepsilon$

$T \to F T'$

$T' \to *FT' \mid \varepsilon$

$F \to (E) \mid id$

$First(E) = First(T) = \{(, id\}$

$First(E') = \{+, \varepsilon\}$

$First(T) = First(F) = \{(, id\}$

First (T') = {*, ε}

First (F) = {(, id}

For follow of the production:

Follow (E) = {), $}

Follow (E') = Follow (E) = {), $}

Follow (T) = First (E') = {First (E') − ε}

∪ Follow (E).

= {+,), $}

Follow (T') = Follow (T) = {+,), $}

Follow (F) = First (T') = {First (T') − ε} ∪

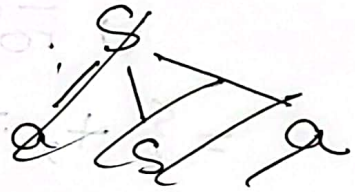
Follow (T) = {*, +,), $}

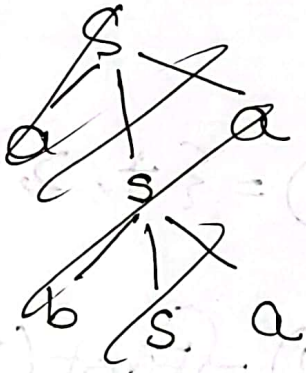
Answer to the Qn.o-3(b)

Ambiguity :- When using the same grammar for the same output string there is more than one parse tree in order to generate that string is called the ambiguity of the grammar. For example-

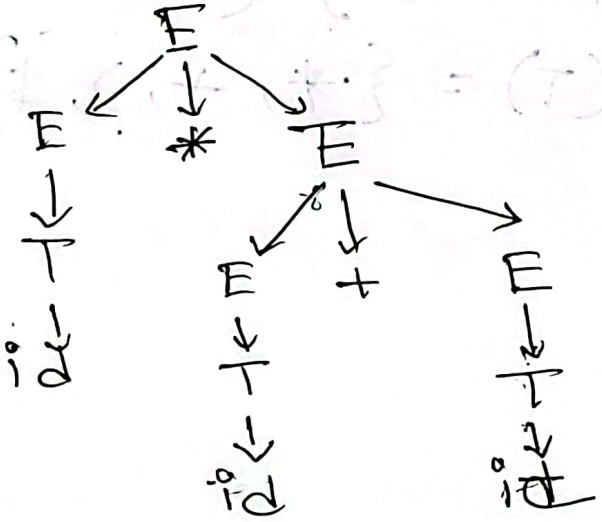
S → aSa
S → b

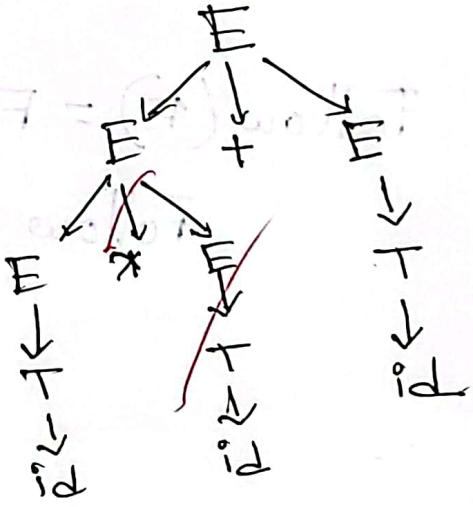
E → E + E | T
E → E * E
T → id





Here,
to generate id * id + id





Here we can see

$E \Rightarrow E+E \mid E*E \mid T$

$T \rightarrow id$

it produces a minimum 2 parse tree in order to generate the string id*id+id

So, this grammar is ambiguous grammar.

In order to do top-down parsing we use the left most derivation of the grammar. But when there is left recursion & the grammar is ambiguos then it is not possible to do top-down parsing. So, we need to remove it.

By using left recurssing removing system

$A \rightarrow A\alpha \mid B$

to

$A \rightarrow BA'$

$A' \rightarrow \alpha A' \mid \varepsilon$

Not Left Recursion

And left factoring removing system

$A \rightarrow \alpha A' \mid \alpha B' \mid c$

to

$A \rightarrow \alpha D \mid c$

$D \rightarrow A' \mid B' \mid \varepsilon$

we can remove ambiguity from the grammar

Owre example- $E \rightarrow E * E$

$E \rightarrow E + E$

$E \rightarrow T$

$T \rightarrow id$

the first 2 lines has left recursion

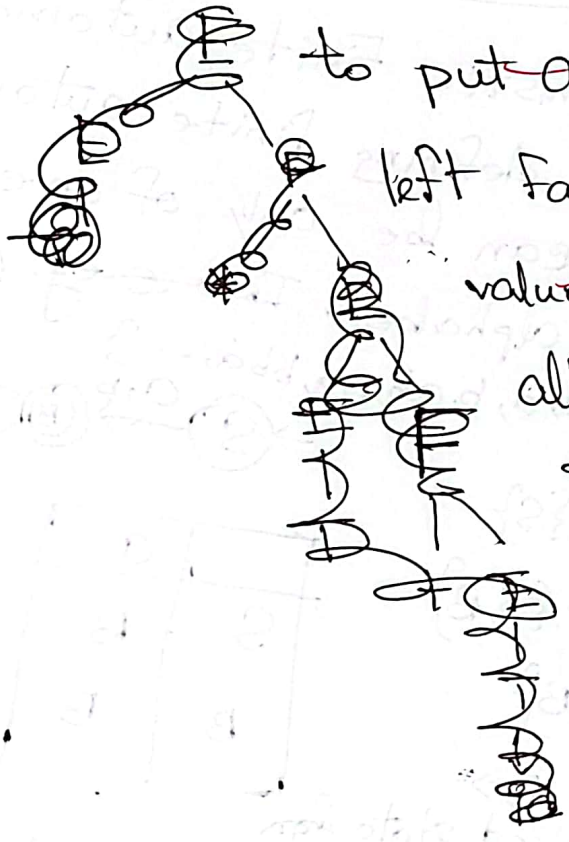
$E \rightarrow E F \mid T$

$F \rightarrow *E \mid +E \mid \varepsilon$

$T \rightarrow id$

Here the left factoring is done.

Farther more we need to put a restriction on the left factoring so that the value does not allows the other production line to change to other line.



Ans to the Qno-4(a)

DFA:- Deterministic Finite automata is a way of ~~dfa~~ defining finite automata where there can be only of one production line for one alphabet. ~~There~~ eg, $(a|b)^*$

$L = \{a, b, ab, aa, bb, a, \ldots\}$

$\Sigma = \{a, b\}$

Starting Symbol; $Q = \{S\}$

No of states; $R = \{S, B\}$

Final state, $\{B\}$

	a	b
S	B	B
B	B	B

Production lines, ; P →

We need to put dead state ~~for~~

NFA:- Nondeterministic Finite automata is the Finite automata where there can be more than one production line for one alphabet.

For a^*b^* the NFA →

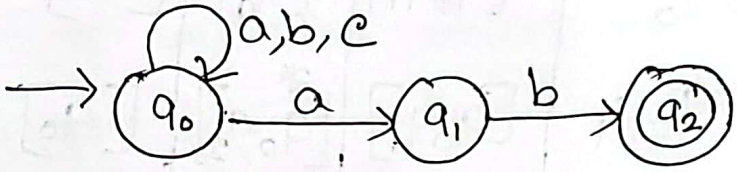
Here for a^*b^* we have more than one production line for a. No need to put dead state.

1938520113

(1)

Answer to the Qno-4 (b).

Given, $(a|b|c)^*ab$.



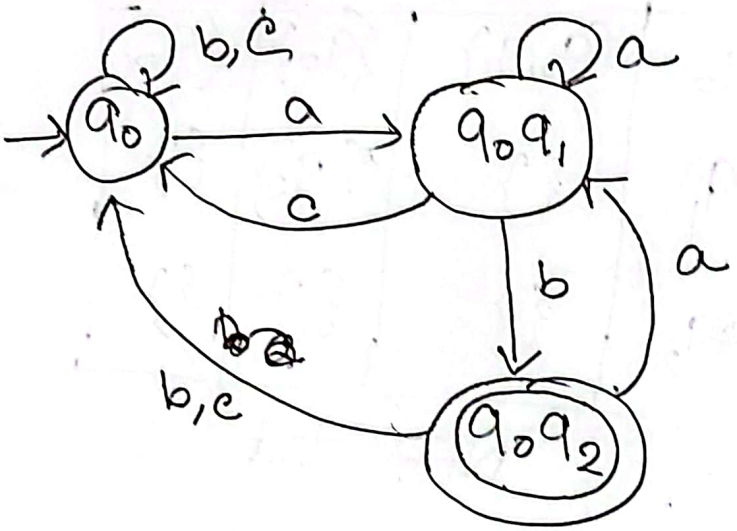
Transition table:

	a	b	c
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\{q_0\}$
q_1	$\{\}$	$\{q_2\}$	$\{\}$
*q_2	$\{\}$	$\{\}$	$\{\}$

NFA

Transition table for DFA:-

	a	b	c
$\rightarrow q_0$	$[q_0 q_1]$	$[q_0]$	$[q_0]$
$q_0 q_1$	$[q_0 q_1]$	$[q_0 q_2]$	$[q_0]$
$*q_0 q_2$	$[q_0 q_1]$	$[q_0]$	$[q_0]$



∴ DFA from NFA

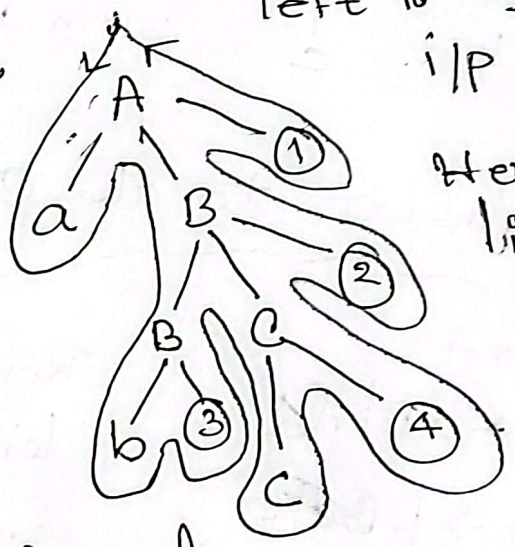
Answer to the Qno-5 @.

Top-down parsing :- The parsing starting with the start symbol and using production line for top down way for determing which production line used for which terminal and using left most derivation of the grammar.

left to right & top to bottom

i/p - abc

A → aB - ①
B → BC - ②
B → b - ③
C → c - ④



Here the first line to derive is ..

o/p :- 3, 4, 2, 1

The production line may or may not use backtracking. If its not using backtracking its called predictive parsing (LL(1)) parser

Bottom-up Parser :- The parser which reads the parse tree from the bottom to top & ~~bottom-up~~ is bottom-up. It compresses the tree from bottom. For the same example

I/P :- abc

O/P :- 3 4 2 1

Here also we read from bottom to top.

Answer to the Q no-5 b)

$S \rightarrow A$

$A \rightarrow aB \mid Ad$

$B \rightarrow bBC \mid f$

$C \rightarrow g$

Remove the left recursion from second line.

(2) 1938520113

$S \to A$

$A \to aBA'$

$A' \to dA' \mid \varepsilon$

$B \to bBC \mid f$

$C \to g$

6.75

First (S) = First (A) = {a}

First (A) = {a}

First (A') = {d, ∈}

First (B) = {b, f}

First (C) = {g}

Follow (S) = {$}

Follow (A) = {$}

Follow (A') = Follow (A) = {$}

Follow (B) = First (C) = {g} ∪ First (A')

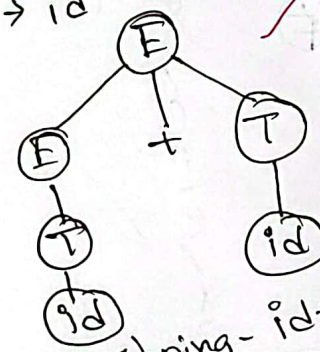
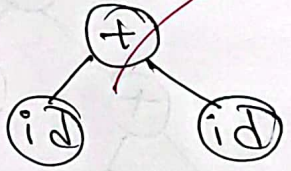
~~Follow (C) = Follow (B)~~ = {g} ∪ {First (A') − ∈} ∪ Follow (A)

= {g, d, $}

Follow(C) = Follow(B) = {g, d, $}

	a	d	b	f	g	$
S	S→A.					
A	A→aBA'					
A'		A'→dA'				A→∈
B			B→bBC	B→f		
C					C→g	

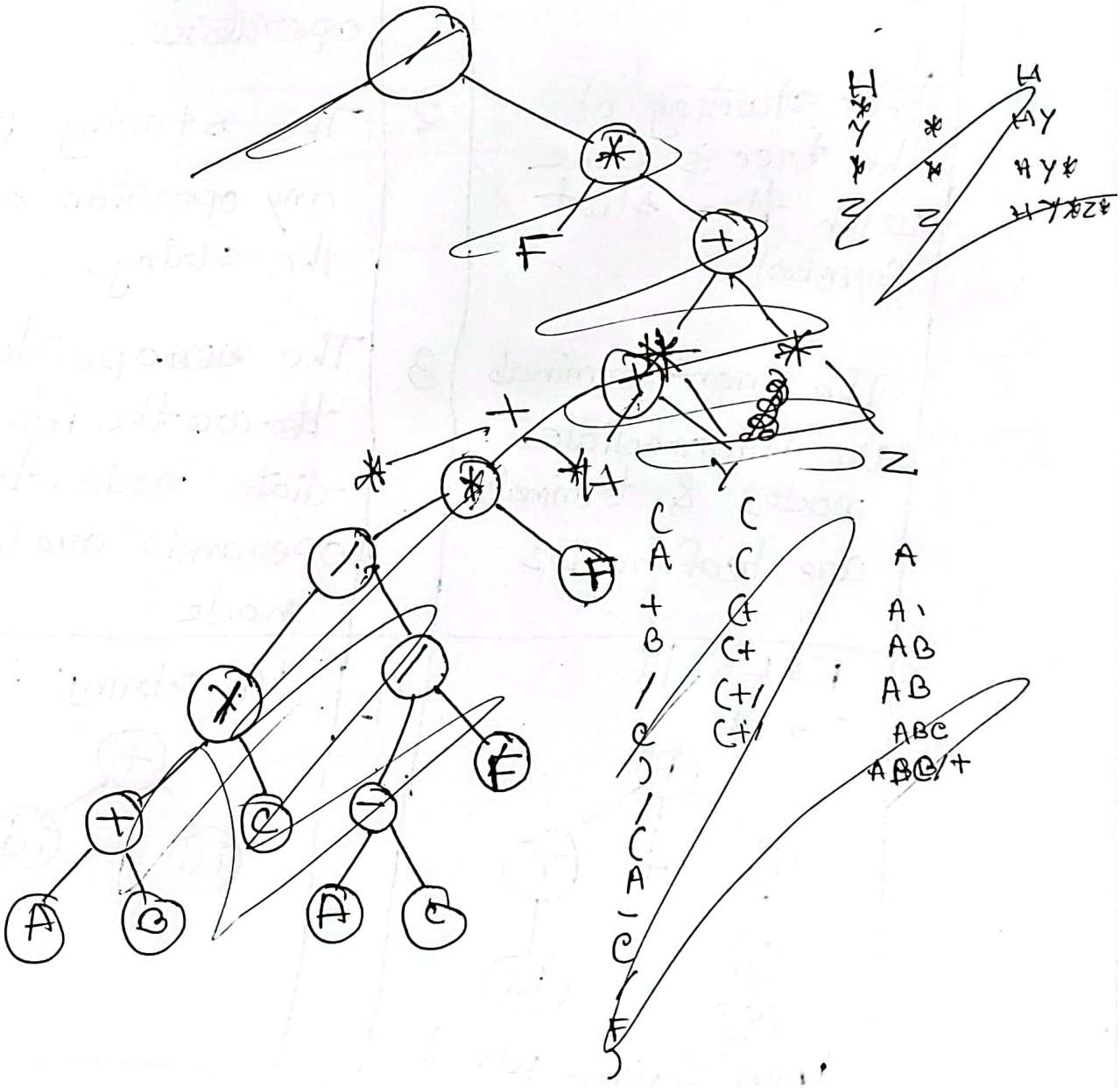
Answer to the Qno-6(a).

Parse tree	No	Syntax tree
The tree which is drawn using the grammar.	1.	The tree which is created by using the operands & operators.
The starting of the tree is done with the start symbol	2	The starting is any operator of the string
The non-terminals are intermediate nodes & terminals are leaf nodes	3.	The operators are the intermediate node. And operands are leaf node
E→E+T|T T→id  Input string- id+id	4	i/p string: id+id 

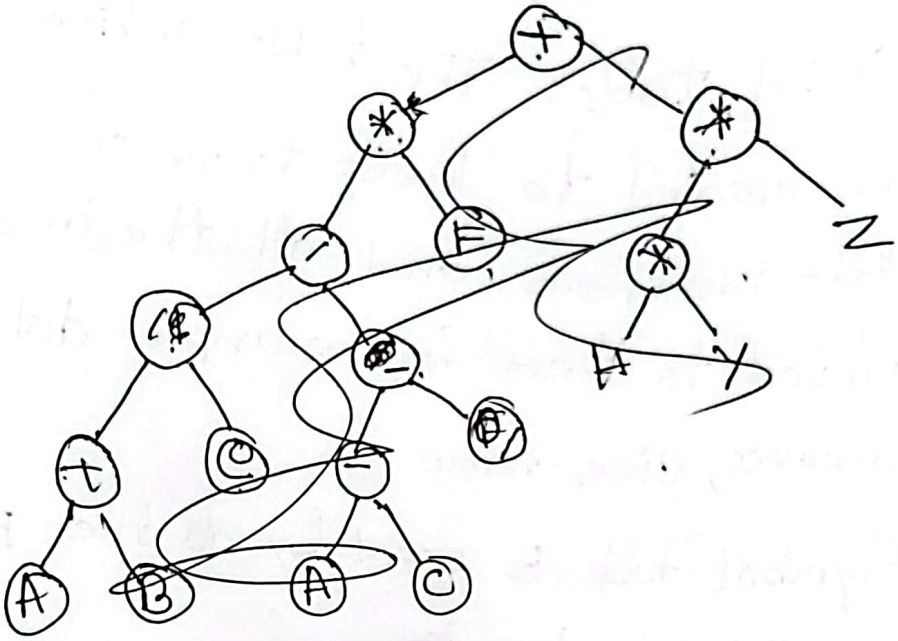
Answer to the Qno-6(b)

Given, $(A+B|C)|.(A-C|F)*F+(H*Y**Z)$

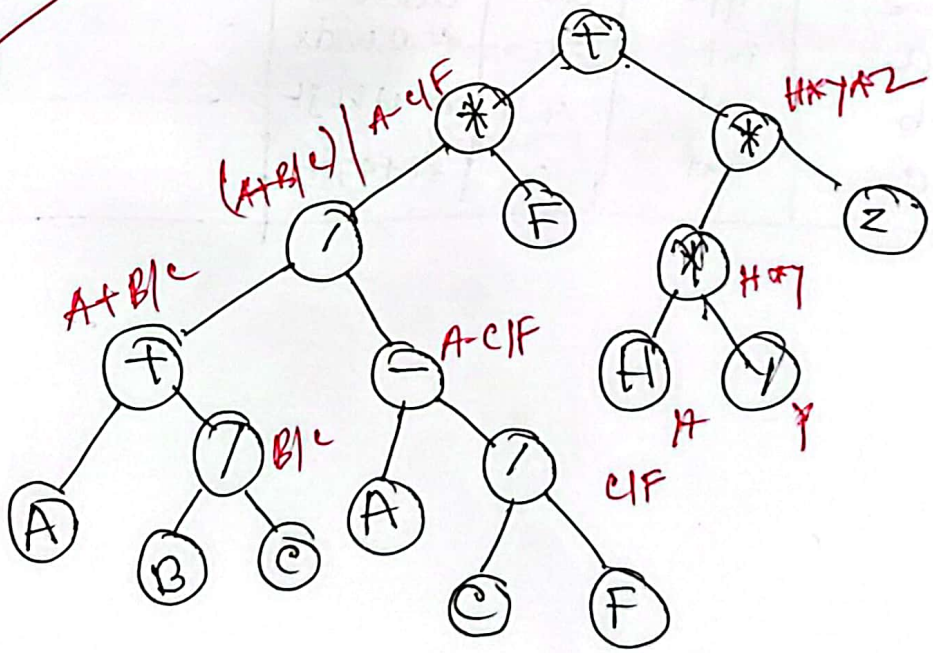
The syntax tree:-



(3) - 1938520113



Syntax tree



Ans to the Qno-6(c).

Symbol table: The table which is created to keep track of identifiers and all the informations related to them for example - data type address, size, name etc.

Symbol table is used by all the phases of the compiler.

```
int a, b, c;
```

Identifier Name	Type	Size	Memory address	
a	int	4	#2bcdx	- - -
b	int	4	#4dkkjt	———
c	int	4	#+4782	

Ans to the Qno-7 (a)

Syntax-directed transition or SDT is used by the Intermediate code generation phase to design the transition for the code that is to be generated.

There are 2 types of SDT,

S-attributed and L-attributed.

S-attributed uses synthesized data and L-attributed uses synthesized and inherereted data (from parent & left sibling only).

Three-address Code:

During intermediate code generation the data are to be stored is a systematic approach, In 3address code- there are use of 3 variables for the calculation

Eg: $id_1 = id_2 + id_3$

$temp_1 = temp_2 + temp_3$

$id_2 = 5.$

Here at most 3 identifier can be used in the operation.

provide five category

1938520113

(4).

Ans to the Qno- 7 (B).

A = B*(C+D).

Quadraple representation:-

operator

$temp_1 = B$

$temp_2 = C + D$

$temp_3 = temp_1 * temp_2$

$A = temp_3$

Quadraple representation:

Operator	Operand1	Operand2	Result
:=	B		$temp_1$
+	C	D	$temp_2$
*	$temp_1$	$temp_2$	$temp_3$
:=	$temp_3$		A

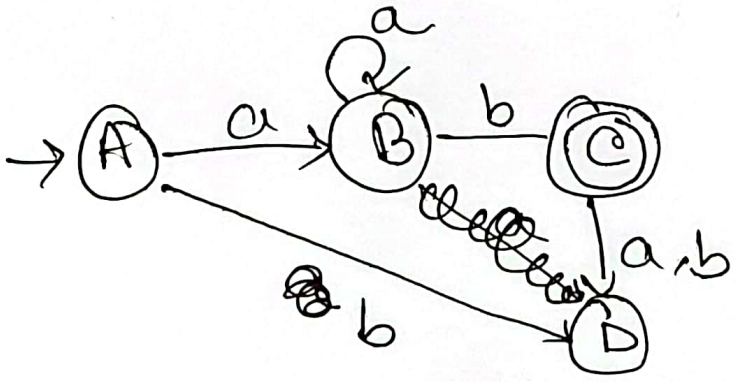
Triple representation

No	Operator	Operand1	Operand2
1	:=	B	
2	+	C	D
3	*	1	2
4	=	3	A

1

Ans to the Qno-7c

Transition diagram: The diagram which is used to represent which data from which state is transform to which state when any specific condition is applied to it. For example the FA (finite automata). Here the data flow from one state to another state is shown with the help of the transition states and the flow from one state to another when we have encountered a specific alphabet. For example



Here is an example of transition diagram of DFA where we have the condition the string will always start with a minimum a and end with only one b.

The flow of the lines are the indicated using the arrow signs.