FRANKLIN'S LECTURES

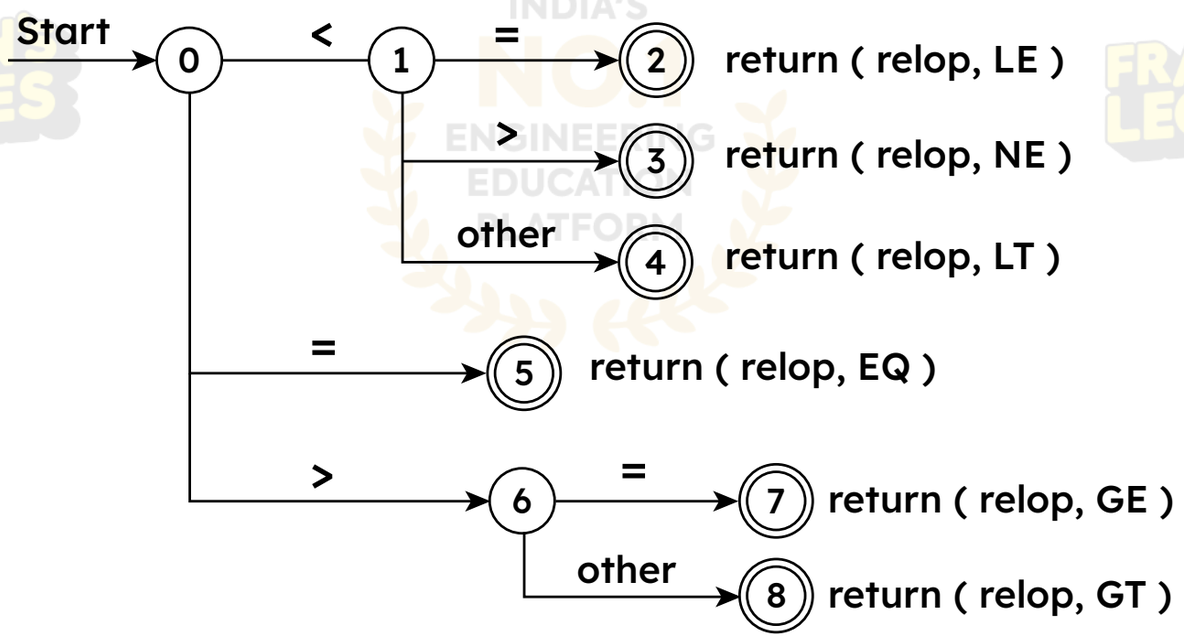
COMPILER DESIGN

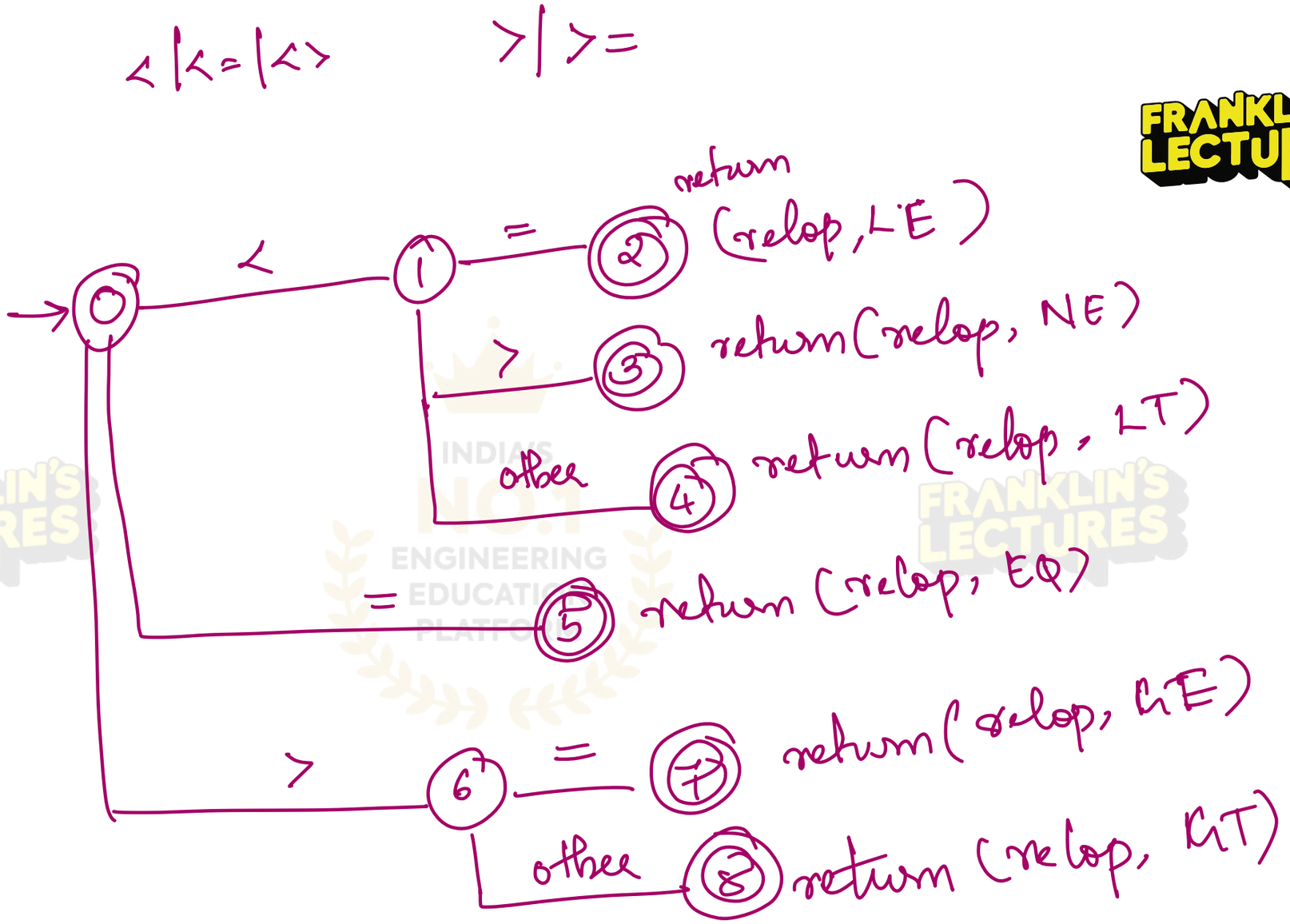
Module 1 | Part 2

CST302

Q1. Draw the transition diagram for the regular definition

relop ⟶ <|<=|=|<>|>=|>




< | <= | <>
> | >=
return
(relop, LE)
return(relop, NE)
return (relop, LT)
return (relop, EQ)
return (relop, GE)
return (relop, GT)
other
other
0
1
2
3
4
5
6
7
8

Q2. With an example source language explain tokens, lexemes & pattem,

Token: Are the terminal symbol in the grammar for the source language. which are generally represented using bold phase names.

Eg:- id, const

lexemes: A lexeme is a sequence of characters in the source pgm that is matched by a pattern for a token

eg: consider the statement, const Pi = 3.14

Here Pi is a lexeme for the token identifier.

Pattern: A pattern is a rule describing a set of lexemes that can represent a particular token in the source program

eg:- pattern for identifier is "letter followed by letters and digits"

Q3. Describe Input buffering scheme in lexical analyzer.

The process of Storing a block of input data in buffer to avoid costly access to secondary storage each time.

Commonly used buffering methods are,

1. One buffer scheme :- Here only one buffer is used to store lexemes. If the the lexeme crosses buffer boundary, the buffer has to be re filled, there by over-writing the buffer.
2. Two buffer scheme: Here two buffers are used to store the input. Here buffers 1 and 2 are scanned alternatively. When the end of the current buffer is reached, the other buffer is filled. Hence the problems that occur in one buffer Scheme is solved.

overwriting.

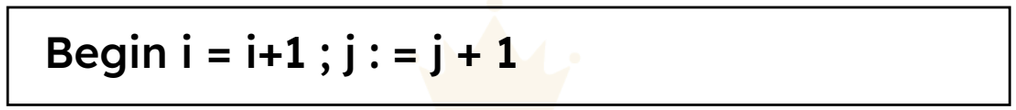


Fig : one buffer scheme

Buffer 1

Fig : Two buffer scheme

Buffer 2

Q4. Construct a regular expression to denote a language L over Σ ={0,1} accepting all strings of 0's and 1's that do not contain Substring 011.

$\Sigma = \{0, 1\}$

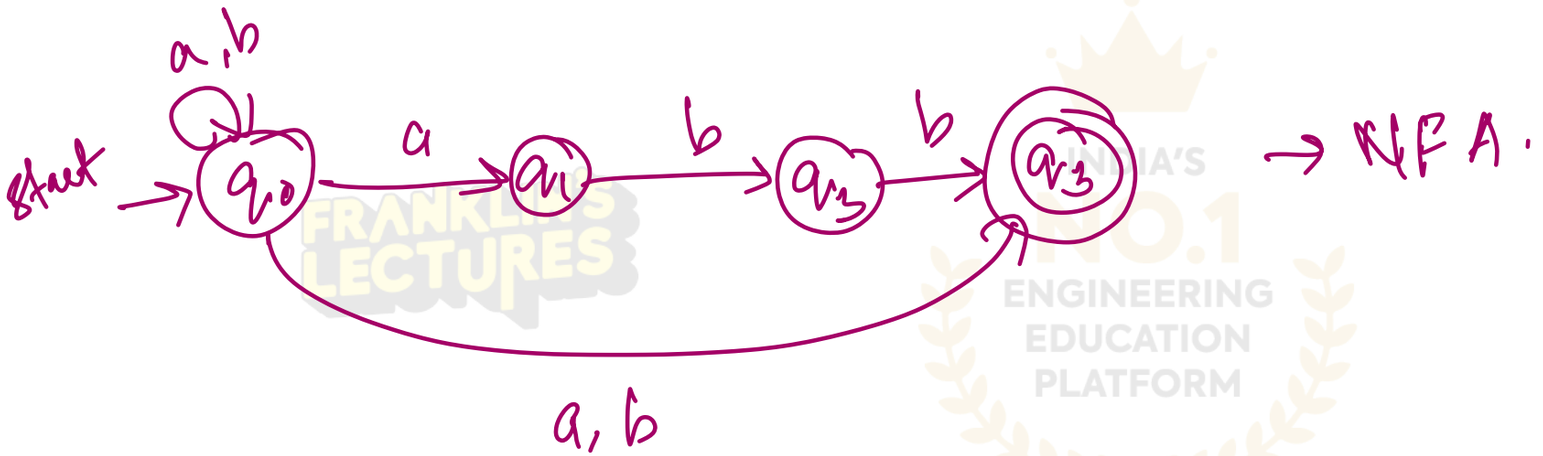
Ans: 1*+1*0 (10+0)* (E+1)

Solution:- construct DFA, then apply Aden's theorem to get regular expression.

$1^* + 1^* 0 (10+0)^* (\epsilon + 1)$

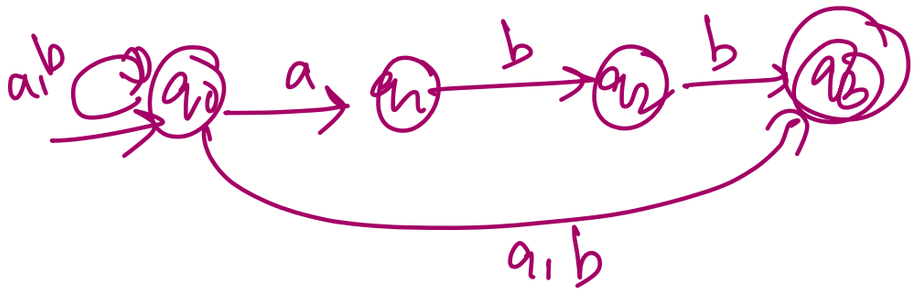
1101 = 110x

Q5. Draw the DFA for the regular expression (a/b)* abb /a + b

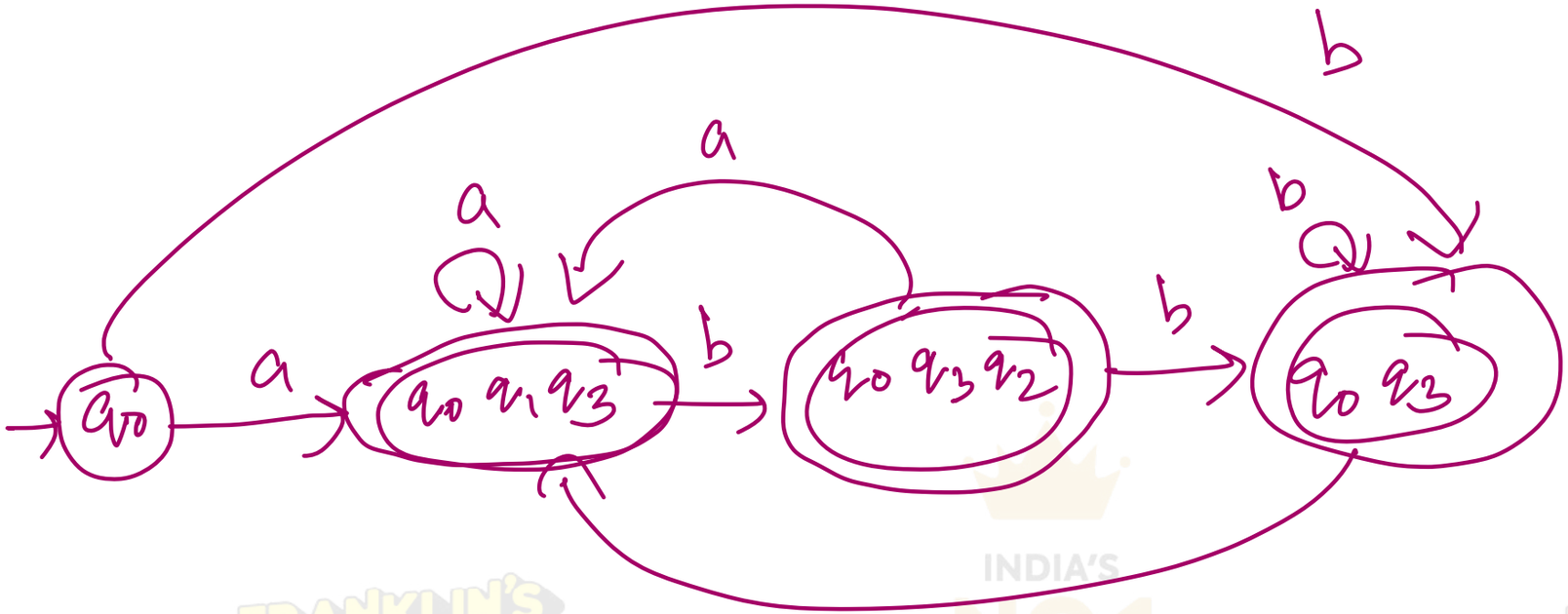


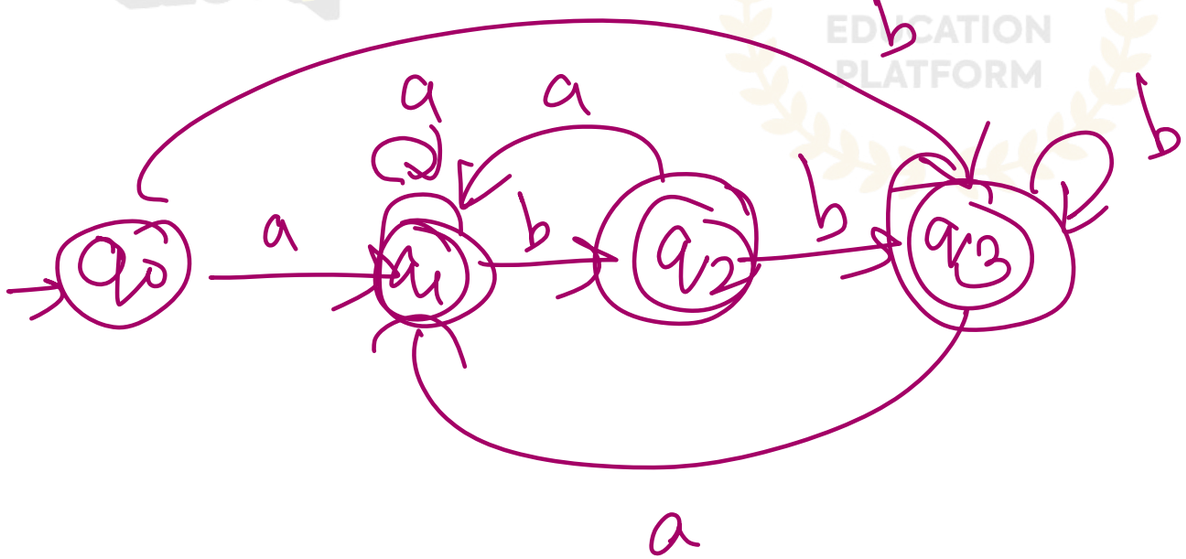
→ NFA.

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM



	a	b
* [q_0]	$q_0\ q_1\ q_3$	$q_0\ q_3$
[$q_0\ q_1\ q_3$]	$q_0\ q_1\ q_3$	$q_0\ q_3\ q_2$
[$q_0\ q_3$]	$q_0\ q_1\ q_3$	$q_0\ q_3$
[$q_0\ q_3\ q_2$]	$q_0\ q_1\ q_3$	$q_0\ q_3$


b
a
a
b
a
b
b
q_0
$q_0\,q_1\,q_3$
$q_0\,q_3\,q_2$
$q_0\,q_3$


a
b
a
a
b
b
q_0
q_1
q_2
q_3

Q6. State the role of lexical analyzer.Identify the lexemes and their corresponding tokens is the following statement

```
Printf("Simple Interest = %f\n, Si);
```

Role of lexical analyzer:-

* Scans the input frons left to right and group the characters into tokens.

Eliminates white spaces and comments

correlates error messages. source programs. with the source program

lexeme: Printf, token: ID,

lexeme: C, token: left paranthesis

lexeme: "Simple Interest = %f\n
token: LITERAL
lexeme: , token : comma.
lexeme:) token: right paranthesis
lexeme: Si, token: ID

lexeme:), token: right paranthesis

lexeme: ;, token: semicolon;

Q7. Explain any three tools that help a Programmer in building a compiler efficiently.

compiler construction tools.

Explain about any three compiler construction tooks. (Refer QnNo.9)

FRANKLIN'S LECTURES
NO.1 ENGINEERING EDUCATION PLATFORM

Q8. Explain the different phases in design of compiler.

1) Lexical analysis
2) Syntactic analysis
3) Semantic analysis.
4) Intermediate code generation
5) code optimization
6) code generation.

Explain each phase with an example.

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

Q9. Explain compiler writing tools

construction-

Compiler writing tools are the tools that have been created for automatic design of specific compiler components. These tools use specialized languages and algorithms. The following is a list of compiler construction tools.

1) **Scanner generator :-** They generate lexical analyzers automatically from the language specification written using regular expressions. It generate finite automata to recognize the regular expression. An example of this tool is LEX.

2) **Parser generator: -** They produce syntax analyzers from context free grammars. Many parser generator utilize powerful parsing algorithm that are too complex to be carried out by hand. An example for this tool is YACC (yet another compiler compiler)

3) **Syntax directed translation Engine:-**
These engine have routines to traverse the parse tree & to produce intermediate code. one or more translations are associated with each node of the parse tree.

4) **Automatic code generators: -** These tools convert the intermediate language into machine language using a collection of rules. Here template matching process is used. An intermediate language statement is replaced by its equivalent machine language statement.

5) **Data flow Engines:-** It is used in code Optimization. These tools perfoms good code optimization using "data-flow analysis" means the gathering of information about how values are transmitted from one part of a program to each other part.

Q10. Explain how the regular expressions and finite automata are used for the Specification and recognition of a tokens.

Scanners are special pattern matching processors. For representing Pattern of strings of characters, regular expressions are used. For example the pattern for the token identifier is, letter-followed by letters and digits. This can be represented by using the regular expression given below.

letter (letter / digit)*

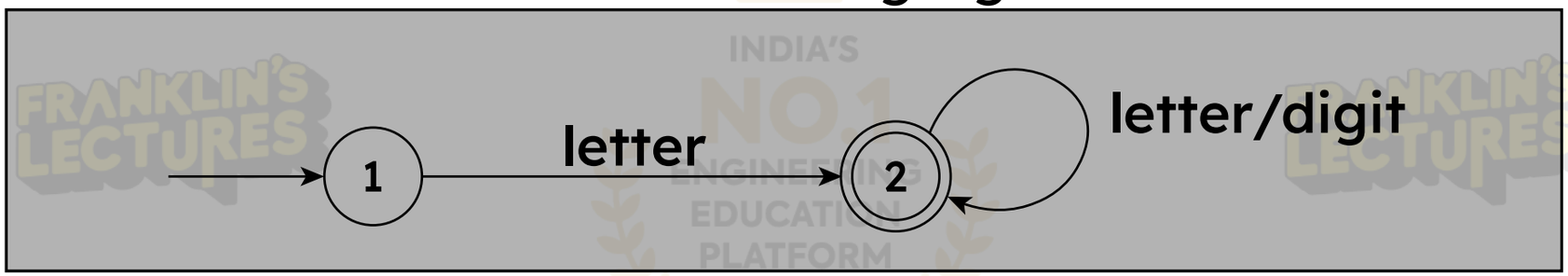
- The tokens obtained during lexical analysis are recognized using a finite automata. Finite automata can be used to describe the process of recognizing patterns in input strings and so they can be used to construct Scanners. A regular definition for an identifier is given below,

letter $\longrightarrow$ a|b|c|.....z|A|B|....|Z

digit $\longrightarrow$ 0|1|2|.....9

identifier $\longrightarrow$ letter(letter/digit)*

- There an identifier consists of a letter followed by any number of letters or digits. The finite automata for the identities can thus be represented as shown in the following figure



finite automata for recognizing an identifier

Q11. Develop a lexical analyzer for the token identifier.

Lexical analyzer convert a sequence of characters into a sequence of tokens. A program that performs lexical analysis may be termed as lexical analyzer/Scanner. The regular definition for an idetifier is given below,

letter → a|b|c|.....z|A|B|....|Z

digit → 0|1|2|.....9

identifier → letter(letter/digit)*

Regular expression for identifier token is Letter (letter/digit)*

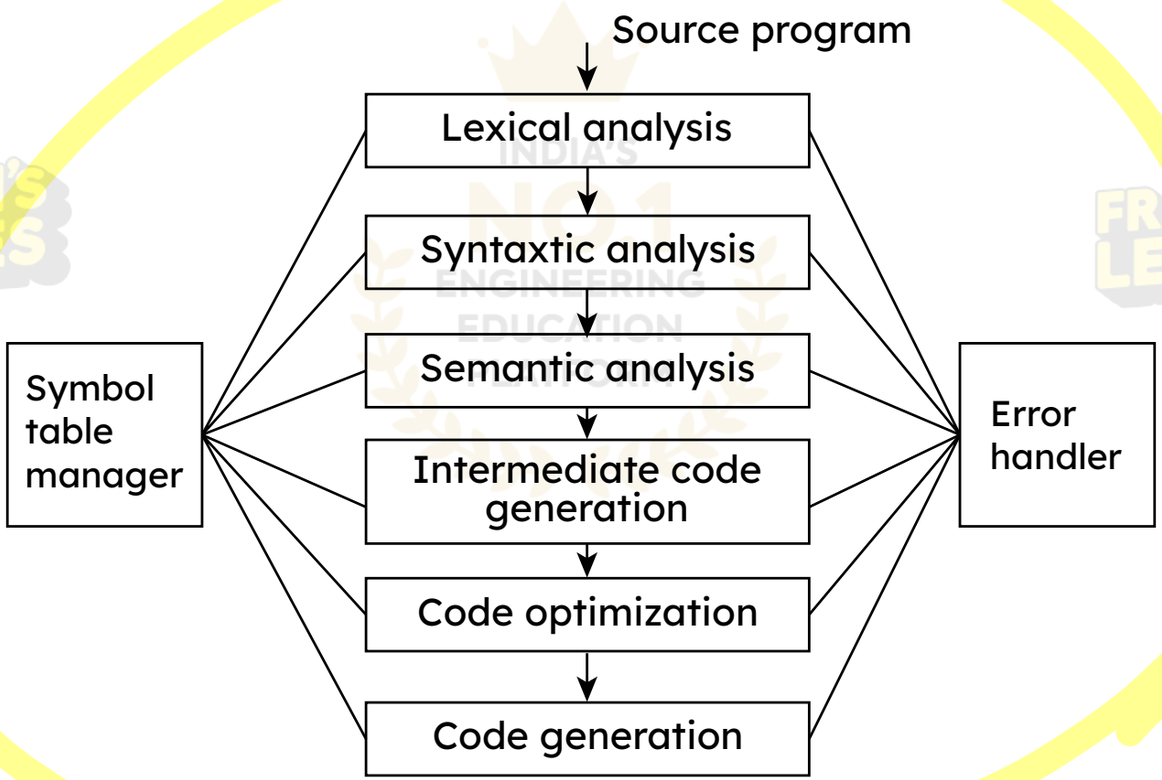
lexical analyzer uses finite automata to recognize the regular expression. The finite automata used by lexical analyzer to identify the idenfier token is given below.

Q12. Trace the output after each phase of the compiler for the assignment statement `a = b+c*10`, if variables given are of float type.

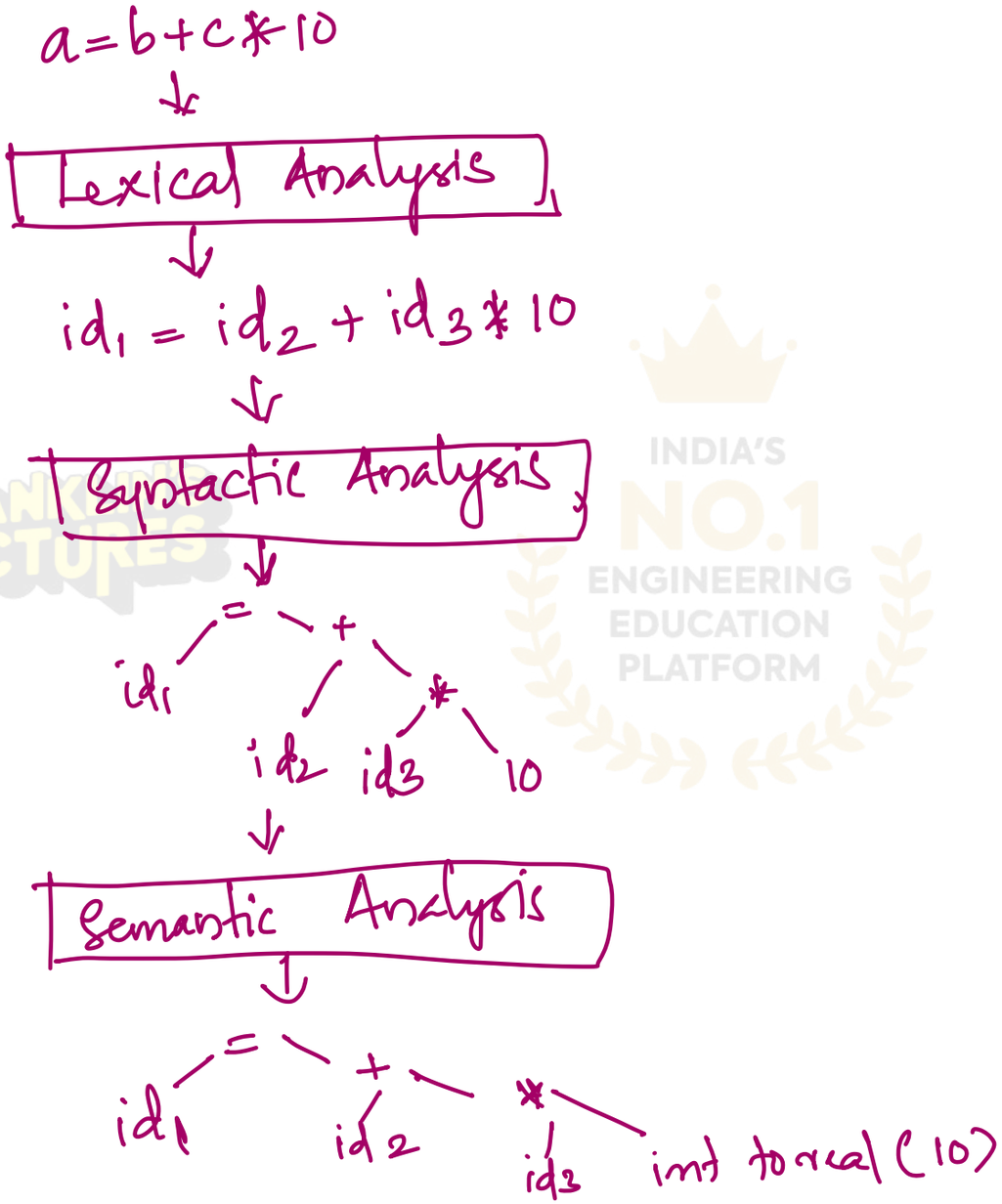
a, b, c — variable

└ float type.

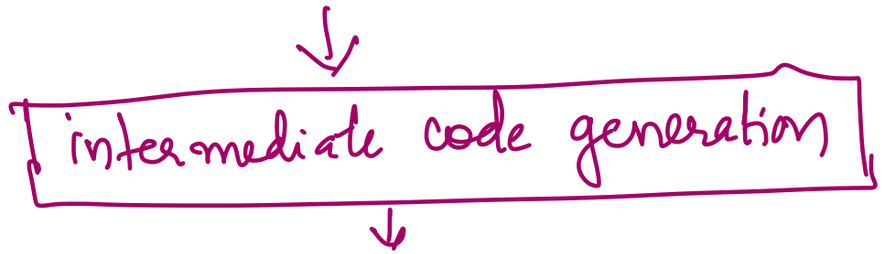
Phases of compiler



$a = b + c * 10$



FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM



temp1 = int to real (10)

temp2 = id3 * temp1

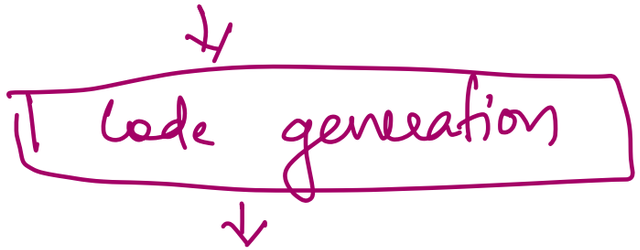
temp3 = id2 + temp2

id1 = temp3

temp1 = id3 * 10.0

id1 = id2 + temp1

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM

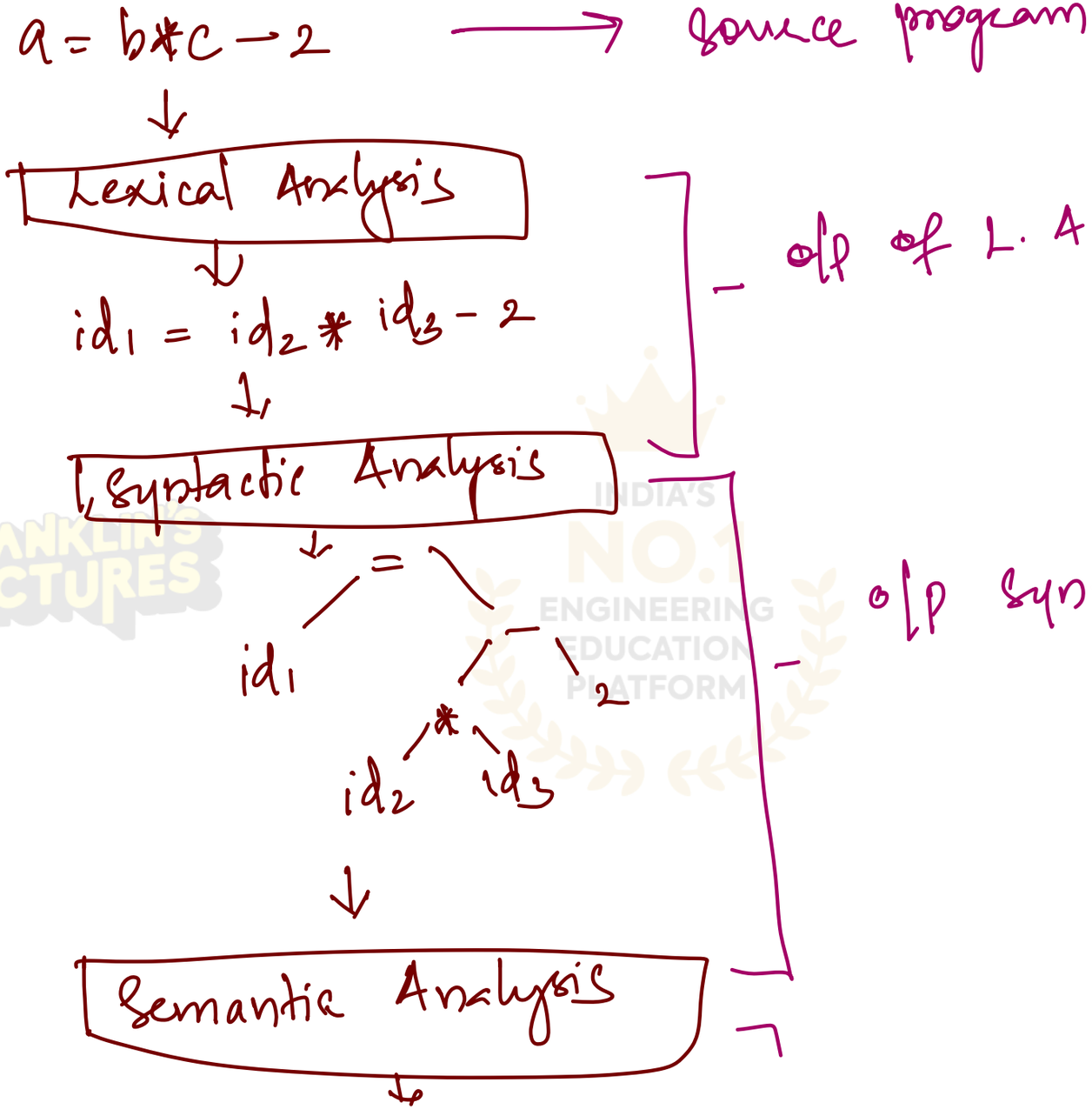


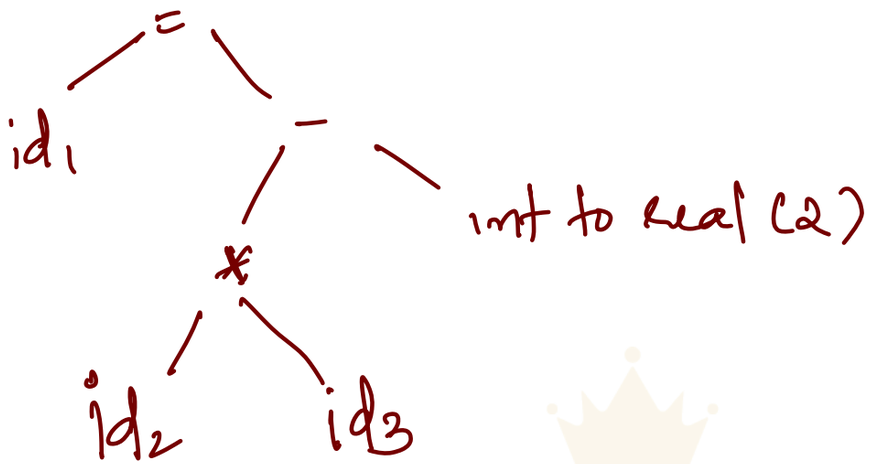

```
MOVF   id3 , R2
MULF   #10.0, R2
MOVF   id2, R1
ADDF   R2, R1
MOVF   R1 , id1
```

FRANKLIN'S LECTURES
INDIA'S NO.1 ENGINEERING EDUCATION PLATFORM
FRANKLIN'S LECTURES
FRANKLIN'S LECTURES

Q13. For a source language statement, a = b*c - 2 where a, b, c are float Variables, * and - represents multiplicatios and subtraction on same data type, show the input & output for each of the compiler Phases.

FRANKLIN'S LECTURES
INDIA'S NO. 1 ENGINEERING EDUCATION PLATFORM
FRANKLIN'S LECTURES


a = b*c - 2
source program
Lexical Analysis
id1 = id2 * id3 - 2
o/p of L. A
Syntactic Analysis
=
id1
-
2
*
id2
id3
o/p Syn. A.
Semantic Analysis

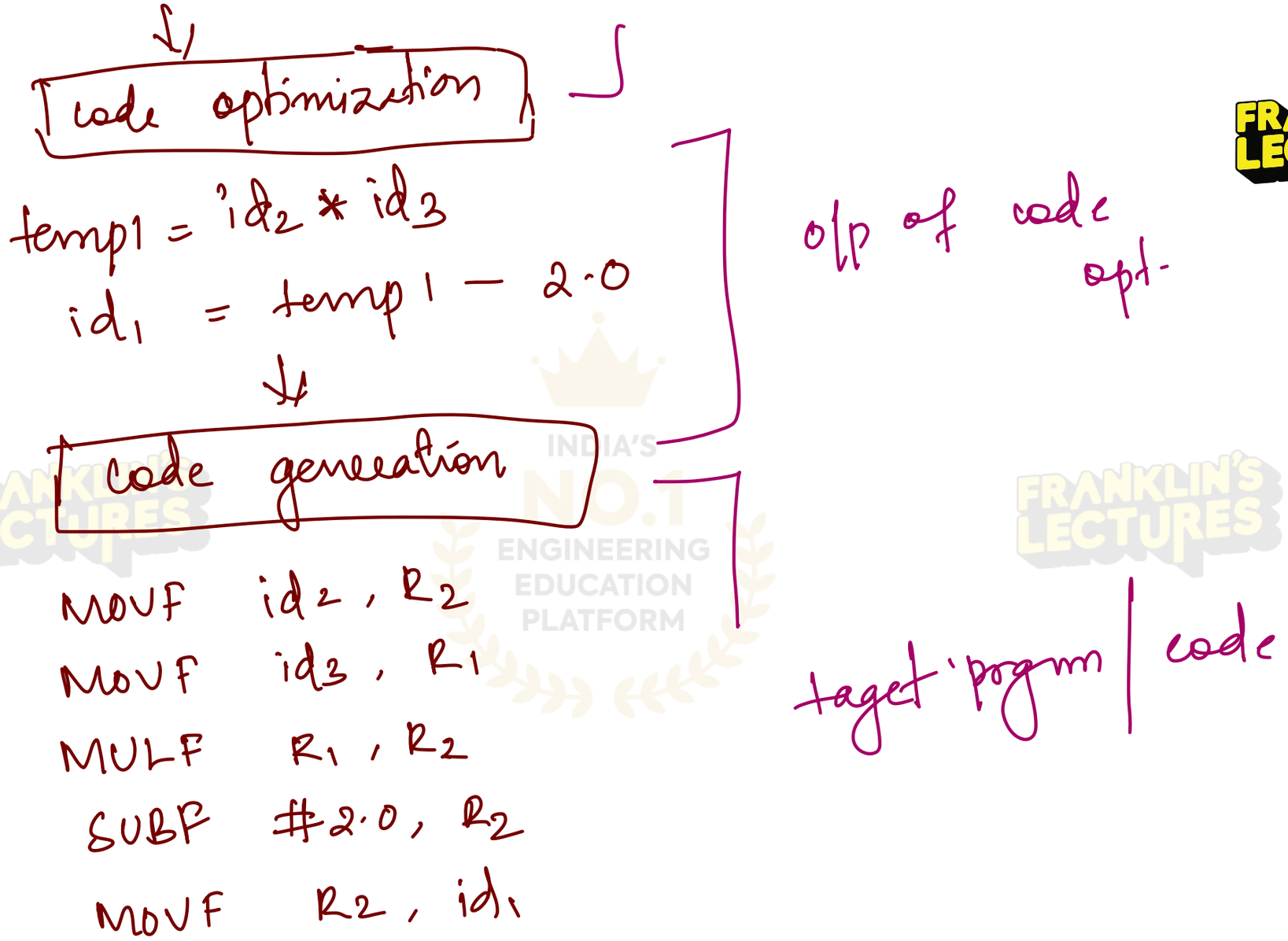


Sem. A o/p

intermediate code generation

```
temp1 = int to real (2)
temp2 = id2 * id3
temp3 = temp2 - temp1
id1 = temp3
```

o/p of ICG.



FRANKLIN'S LECTURES
INDIA'S N.O.1 ENGINEERING EDUCATION PLATFORM

THANK YOU