Name: Akashdeep Soni

Roll NO: 1816410036   23/09/20

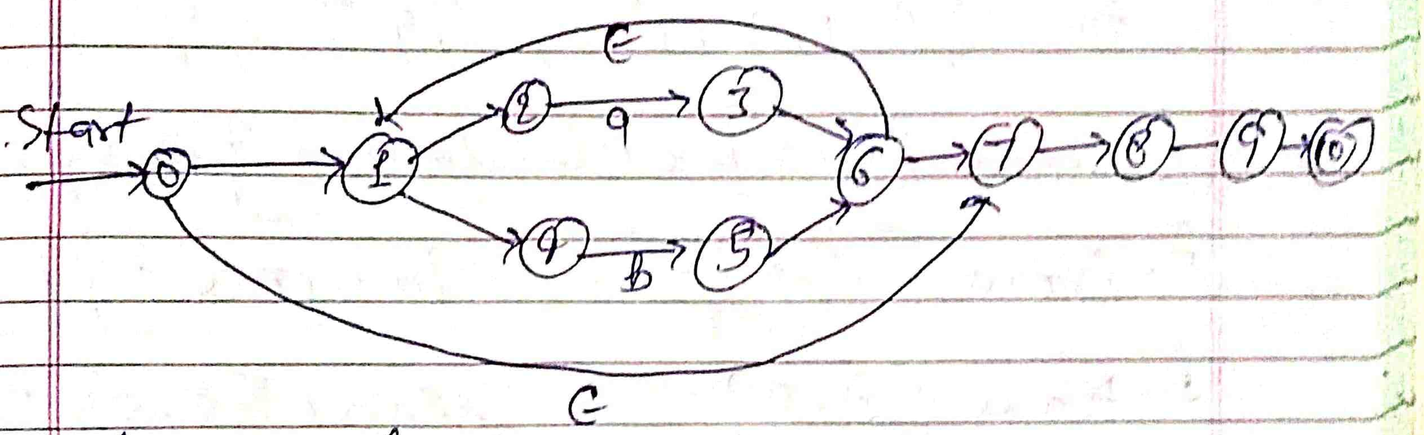
Branch & Section: CS 3B

Subject: Compiler Design (KCS-502)

Assignment → 1

Answer → 1

NFA with ε-moves



A = ε-closure (0)
= {0, 1, 2, 4, 7}

* Dtrans [A, a] = ε-closure {move (0, a) U move (1, a) U move (2, a) U move (4, a) U move (7, a)}
= ε-closure {3, 8}
= {3, 6, 7, 1, 2, 4, 8} = B

* Dtrans [A, b] = ε-closure {move (0, b) U move (1, b) U move (2, b) U move (4, b) U move (7, b)}
= ε-closure {5}
= {6, 7, 5, 1, 2, 4} = C

* Dtrans [B, a] = ε-closure {move (3, a) U move (6, a) U move (7, a) U

move (1, a) · U move (2, a) U move (4, a)
U move (8, a) }
= ε-closure { 8, 3 } = B

* D trans [B, b] = ε-closure { move (3, b)
U move (6, b) U move (7, b) U move (1, b)
U move (2, b) U move (4, b) U move (8, b)
}
= ε-closure { 5, 9 }
= { 6, 1, 2, 4, 5, 7, 9 } = D

* D-trans [C, a] = ε-closure { 8, 3 } = B

D-trans [C, b] = ε-closure { 5 } = C

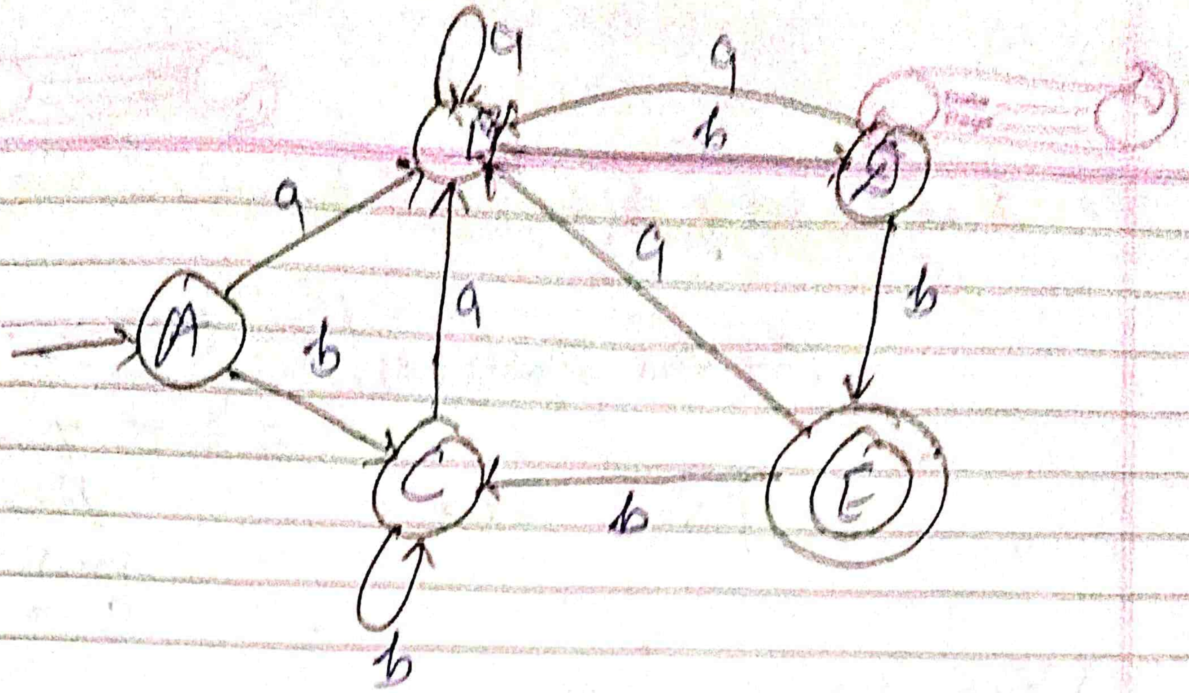
D-trans [D, a] = ε-closure { 3, 8 } = B

D-trans [D, b] = ε-closure { 5, 10 } = E
= { 7, 6, 1, 2, 4, 5, 10 }

D trans [E, a] = ε-closure { 8, 3 } = B

D trans [E, b] = ε-closure { 5 } = C

States	a	b
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C



Answer → 2

Phases of Compiler:

1. Lexical Analyzer:

* Lexical analysis phase converts source program into tokens with the help of Regular expression. RE for identifier = $l(l+d)^*$

* Meaningful character stream is called lexems and for each lexeme, we generate tokens [Token = <token name, token no>]

2. Syntax Analysis phase:

Famous Syntax Analyzer tool (Yet another compiler compiler)

<id,1> <=> <id,2> <+> <id,3> <*> <60>

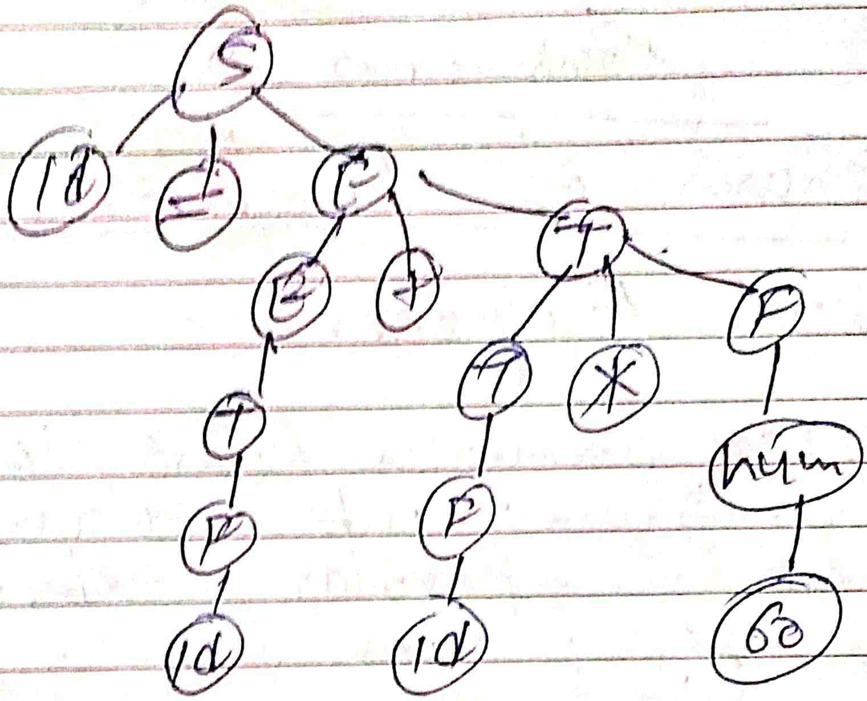
↓

Syntax Analyzer ← CFG → E → E+E | T
T → T*F | F
F → id | num
num → 0 | ... 9
S → id = E

↓

Parse Tree

Now parse tree



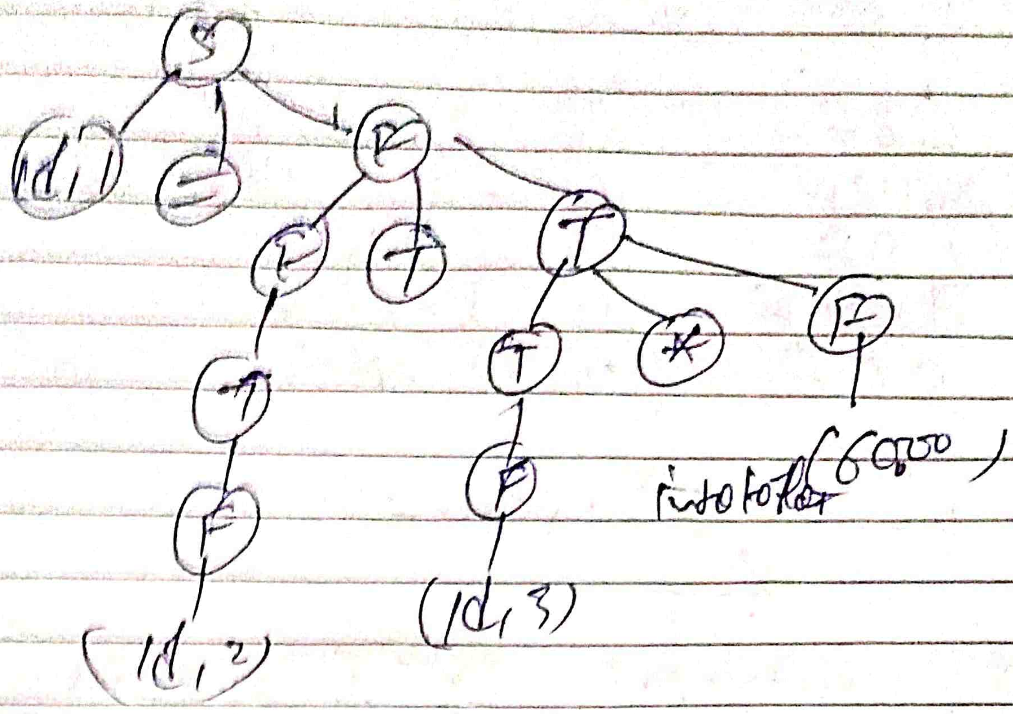
It takes input as a stream of tokens with the help of the grammer we generate parse tree for these tokens.

3. Semantic Analysis: Type checking
Type Conversion

int a
float b
b = a;
Implicit

int a
floa b
a = (int) b
Explicit

Semanticly verified parse tree.



1 Intermediate Code generation:

3 address code

$t_1 = 60.00$

$t_2 = (id,3) * t_1$

$t_3 = (id,2) * t_2$

$(id,1) = t_3$

II. immediate addressing mode

(a) Indirect addressing mode

(b) Code Generation: Optimizing the code by putting constants directly using immediate addressing mode

$t_2 = (id,3) * \langle 60.0 \rangle$

$(id,1) = (id,2) + t_2$

⑥ Target Code generation

Generate code

```
LDF    R1 , (id,3)
MULF   R2 , R1 , #60.0
LDF    R3 , (id,2)
ADDF   R3 , R3 , R2
STF    (id,1) , R3
```

Answer → 3

Minimising DFA means reducing the numbers of states from given DFA.

2 states P & Q are equivalent when

$\delta(p, w) \in F \Rightarrow \delta(q, w) \in F$

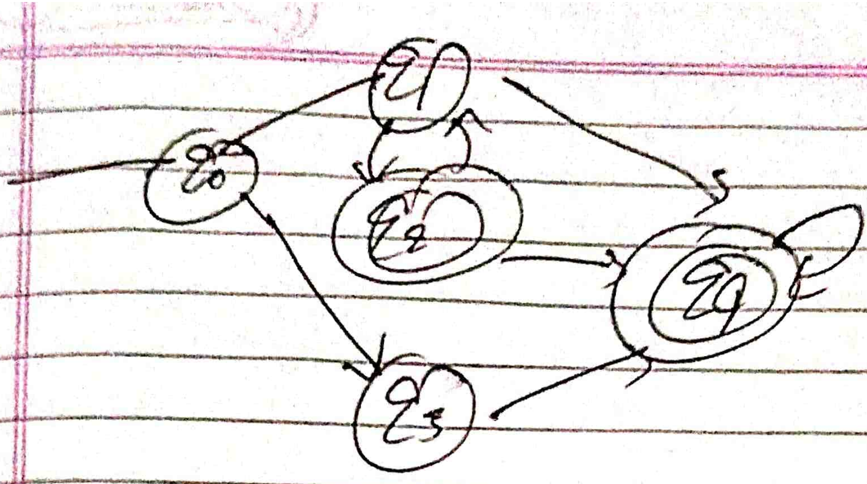
$\delta(p, w) \notin F \Rightarrow \delta(q, w) \notin F$

We have to various steps

Step1 Delete all the states from given DFA which are not reachable from initial state

Step2 Draw the transition table

Step3 Divide final and non-final states

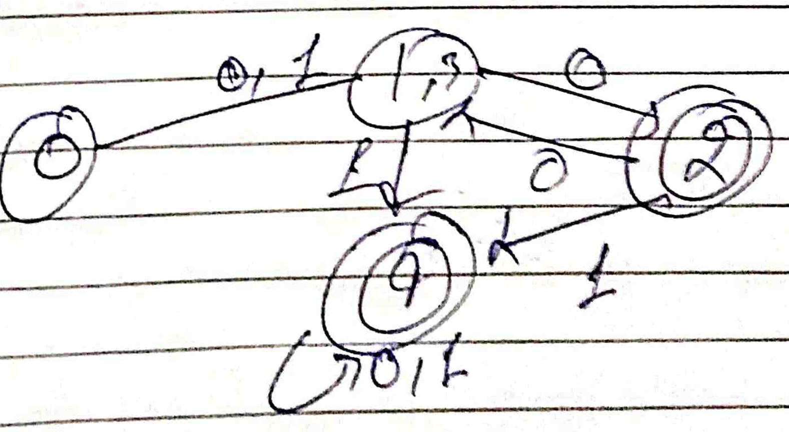


$t_0 = \{q_0, q_1\ q_3\}\ \{q_2, q_4\}$

$q_1 = \{q_0\}\ \{q_1, q_3\}\ \{q_2\}\ \{q_4\}$

< Transition Table

δ	0	1
q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
q_4	q_4	q_4


0,1
1,3
0
0
2
1
1
0
4
0,1