Types [based on 44th batch]

No	Topics
1	1. Theory Only
2	1. Consturct Automata LR(0) for following grammar 2. Given grammar construct FIRST and FOLLOW , LL(1) , Predictive parser 3. Convert CFG to CNF
3	1. Given NFA construct DFA 2. Minimize DFA 3. Make grammar suitable for top-down parsing 4. Given table show moves of LR parser
4	1. Design CFG using logical operators 2. Construct regular expression 3. Design PDA 4. Construct DFA for regex 5. Write down the three address code

Qubits-45 Suggestion (By Joy Sir)

No	Topics
1	Theory Only (Answer 5 Question out of 7 Question => 4*5=20)
2	Simulation Based / Design (Answer 2 Question out of 3 Question => 2*7=14)
3	Simulation Based / Design (Answer 3 Question out of 4 Question => 3*4=12)
4	Theory + Design (Answer 4 Question out of 5 Question => 4*6=24)
	Must Solve: Previous Semester + Quiz + Book Exercise
	For Theory: 1. Conceptual Theory **2. Parsing এর সাথে Formal Language এর কোথায় কাজে লাগে।**

Decipher-44 Final

Set 1

Question 1. Answer any five (5) questions. [Marks: 5x4=20]

a) Briefly explain a Language Processing System. [4]

b) Write short notes on Regular Expression Operators and mention the order of precedence of those operators. [4]

c) What is a Finite Automaton? How Deterministic Finite Automata is different from Nondeterministic Finite Automata? Explain with appropriate examples. [4]

d) Explain the concept of Syntax Directed Translation and Attribute Grammar. [4]

e) Illustrate the common error recovery techniques in parsing. [4]

f) Describe the preliminary simplifications that are needed to be made to get a normal form of CFG. [4]

g) Explain two representations of the three-address code with appropriate examples. [4]

Set 2

Question 2. Answer any two (2) questions. [Marks: 2x7=14]

a) Construct the LR(0) automaton for the following grammar: [7]

$$S \rightarrow aXd$$
$$X \rightarrow YZ$$
$$Y \rightarrow b$$
$$Z \rightarrow c$$

b) Consider the grammar, G and answer the following questions:

$$S \rightarrow AcBd$$
$$A \rightarrow bA \mid \varepsilon$$
$$B \rightarrow cBd \mid a$$

i. Find FIRST and FOLLOW sets. [3]
ii. Construct the LL(1) parsing table. [2]
iii. Show the moves a predictive parser makes on input "ccad". [2]

c) Convert the following Context Free Grammar (CFG) to Chomsky Normal Form (CNF).

$$S \rightarrow ASA \mid aB$$
$$A \rightarrow B \mid S$$
$$B \rightarrow b \mid \varepsilon$$

Set 3

Question 3. Answer any three (3) questions. [Marks: 3x4=12]

a) Construct a DFA equivalent to the following ε-NFA. [4]

	ε	a	b
→ *1	{3}	∅	{2}
2	∅	{2, 3}	{3}
3	∅	{1, 3}	∅

b) Minimize the following DFA. [4]

	0	1
→ a	c	b
*b	d	d
c	a	b
*d	d	d

c) Make the following grammar suitable for top-down parsing. [4]

$S \rightarrow (L) \mid a$
$L \rightarrow L, S \mid S$

d) Considering following grammar and parsing table, show the moves of an LR parser on **id+id**. [4]

$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$

State	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		S6				acc			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

Set 4

Question 4. Answer any four (4) questions. [Marks: 4x6=24]

a) Considering the following examples, design a CFG for Expressions using <, >, <=, >= logical operators and +, - arithmetic operators. Also derive any of the example strings given below with your CFG. Here, **Σ = {a, b, c, <, >, <=, >=, (,), +, -}** [6]

Examples:

- a < b
- b - c
- (a + b) < c
- (a + b – c) <= (b + c – a)
- (a < b) + (c >= b)

<a> →
<a 0> →
<sym> →

b) Construct Regular Expressions for the following languages over alphabet {0, 1}: [6]

i. L = {w | w starts with **0** but not having consecutive **1**'s}
ii. L = {w | w consists of alternating **0**'s and 1's.}
iii. L = {w | w consists of **0**'s and **1**'s not containing 101 as a substring.}

c) Design PDA for the language, L = {w01w^R | w is in (0+1)*} over alphabet {0, 1} and show the ID's of the PDA for input **100101**. [6]

d) Construct DFAs from the following regular expressions: [6]

i. 01*01*01
ii. 1*01*0(0|1)*

e) Write down the Three Address Code for the following **C** code snippet. [6]

```
1. b=3;
2. a=1+b*3;
3. if(a>b)
4.    a=a-2;
5.    for(;a>b;a--)
6.    {
7.       b=a+b;
8.       func1(b);
9.    }
10.
11. switch(b)
12. {
13.    case 1: a--; break;
14.    case 2: b=-a*3+5;
15.    default: a=0;
16. }
```

Integer-43 Final

Question 1. [Marks: 14]

a) Explain the relationships among Token, Lexeme & Pattern from the perspective of a lexical analyzer. **[4]**

b) Eliminate epsilon-productions from the following grammar and derive ***cdbba*** using the resulting grammar. **[5]**

$$S \rightarrow cAdB$$
$$A \rightarrow Ab \mid \varepsilon$$
$$B \rightarrow Aa \mid \varepsilon$$

c) Construct a DFA for the language of all strings consisting of **0**s and **1**s which start and end with the same symbol. **[5]**

b. By 67 (Bappy)

After eliminating epsilon productions,

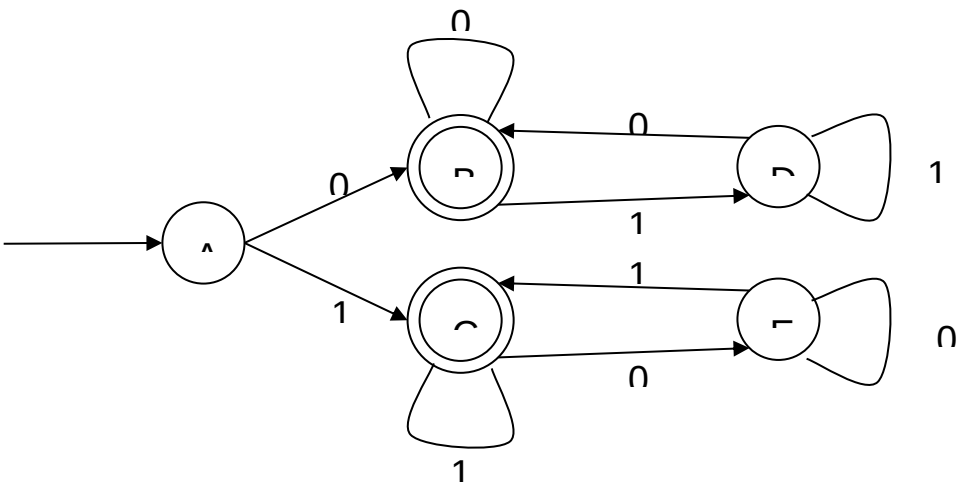
S -> cAdB | cdB | cAd | cd
A -> Ab | b
B -> Aa | a

Derivation of cdbba,

S -> cdB
-> cdAa [Using B -> Aa]
-> cdAba [Using A -> Ab]
-> cdbba [Using A -> b]

[Derived]

c. *by 24*
Regular Expression: 0(0|1)*0 | 1(0|1)*1 | 0 | 1



Question 2. [Marks: 14]

a) Explain the "Power of an Alphabet" with an example and the difference between Σ and Σ^1 where Σ is alphabet. **[3]**

b) Convert the following NFA to an equivalent DFA and indicate the unreachable states from the start state of the resulting DFA. **[5]**

	0	**1**
→ **a**	{a, b}	{a}
b	{c, d}	{c}
c	{a, c}	{e}
***d**	∅	∅
***e**	∅	∅

c) Construct Regular Expressions for the following languages over alphabet {A, B}: **[6]**

i. L = {w | w starts with **A** but not having consecutive **B**s}

ii. L = {w | w contains substrings of "at least two **A**s" and "three **B**s"}

iii. L = {w | w consists of **A**s and **B**s in which two **B**s do not come together}

a. by 24

Powers of an Alphabet:

- If Σ is an alphabet, the set of all strings of a certain length from that alphabet can be expressed by using an exponential notation.
- Σ^k is defined as the set of strings of length k, each of whose symbols is in Σ.
- $\Sigma^0 = \varepsilon$, no matter what the alphabet Σ is. In other words, ε is the only string of length 0.
- Example: If Σ = {a, b, c} then Σ^1 = {a, b, c}, Σ^2 = {aa, ab, ac, ba, bb, bc, ca, cb, cc}, Σ^3 = {aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc}.

b.

c. **Solved by Younus-131**

i. A (A + BA)* (B + Ɛ)

alternative : A (B + Ɛ) (A + AB)*

ii. (A + B)* AA (A + B)* BBB (A + B)* | (A + B)* BBB (A + B)* AA (A + B)*

iii. (A + BA)* (B + Ɛ) - **Explanation :** https://youtube.com/watch?v=IswH4GqO8Sq&feature=share

alternative : (B + Ɛ) (A + AB)* ***Both are correct***

Question 3. [Marks: 14]

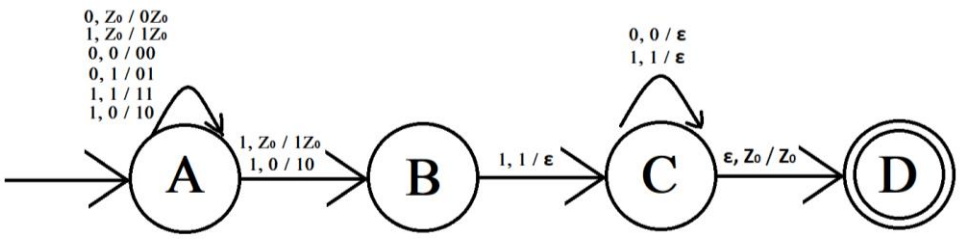
a) Describe the roles of Syntax Analyzer and Semantic Analyzer of a compiler. [4]

b) Remove Unit Productions from the following grammar. [5]

$$S \rightarrow XY$$
$$X \rightarrow a$$
$$Y \rightarrow Z \mid b$$
$$Z \rightarrow M$$
$$M \rightarrow N$$
$$N \rightarrow a$$

c) Design PDA for the language, L = {w11w^R | w is in $(0+1)^*$} over alphabet {0, 1}. [5]

C. **Solved by Younus-131 - ektu check kore nis, ami etai disi**



Question 4. [Marks: 14]

a) For a CFG to be in CNF, some conditions are needed to be met. Describe the conditions in your own words. **[4]**

b) Find FIRST and FOLLOW sets for the following grammar. **[5]**

$$
\begin{aligned}
S &\rightarrow aBDh \\
B &\rightarrow cC \\
C &\rightarrow bC \mid \in \\
D &\rightarrow EF \\
E &\rightarrow g \mid \in \\
F &\rightarrow f \mid \in
\end{aligned}
$$

c) Write down the Three Address Code for the following code snippet. **[5]**

```
1. a=4; b=2;
2. if(a<5)
3.      do
4.      {
5.        if(a==b)
6.        {
7.          b=a+b;
8.          func1(a, b);
9.        }
10.     } while(a--);
11. switch(b)
12. {
13.    case 1: a--;
14.    case 2: b=-a*3+5; break;
15.    default: a=0;
16. }
```

```
a = 4
b = 2
if a<5 goto A
goto B

A: if a==b goto C
   goto D

C: t1 = a+b
   b = t1
   param a
   param b
   call func1, 2

D: if a==0 goto E
   t1 = a-1
   a = t1
   goto A

E: t1 = a-1
   a = t1

B: if b==1 goto F
   if b==2 goto G
   goto H

F: t1 = a-1
   a = t1

G: t1 = -a
   t2 = t1*3
   t1 = t2+5
   b = t1
   goto Z

H: a = 0

Z: End
```

a. By 67 (Bappy)

For a CFG to be in CNF, at first following 3 conditions are needed to met-

1. No epsilon-productions in the grammar
2. No unit productions in the grammar
3. No Useless symbol in the grammar

Then all the production of the grammar should be in one of two following simple forms-

1. A -> BC, where A, B and C, are each variables OR
2. A -> a, where A is a variable and a is a terminal.

b. **Solved by Younus-131**

First(S) = {a} **Follow(S) = {$}**

First(B) = {c} **Follow(B) = {g, f, h}**

First(C) = {b, Ɛ} **Follow(C) = {g, f, h}**

First(D) = {g, f, Ɛ}	**Follow(D) = {h}**
First(E) = {g, Ɛ}	**Follow(E) = {f, h}**
First(F) = {f, Ɛ}	**Follow(F) = {h}**

c. **Solved by Younus-131**

Question 5. [Marks: 14]

a) Explain all the components of a Context Free Grammar (CFG) with an example and describe the importance of CFG in compiler design. **[4]**

b) Minimize the following DFA. **[5]**

	0	1
→ *a*	*b*	*c*
b	*a*	*c*
**c*	*d*	*d*
**d*	*d*	*d*

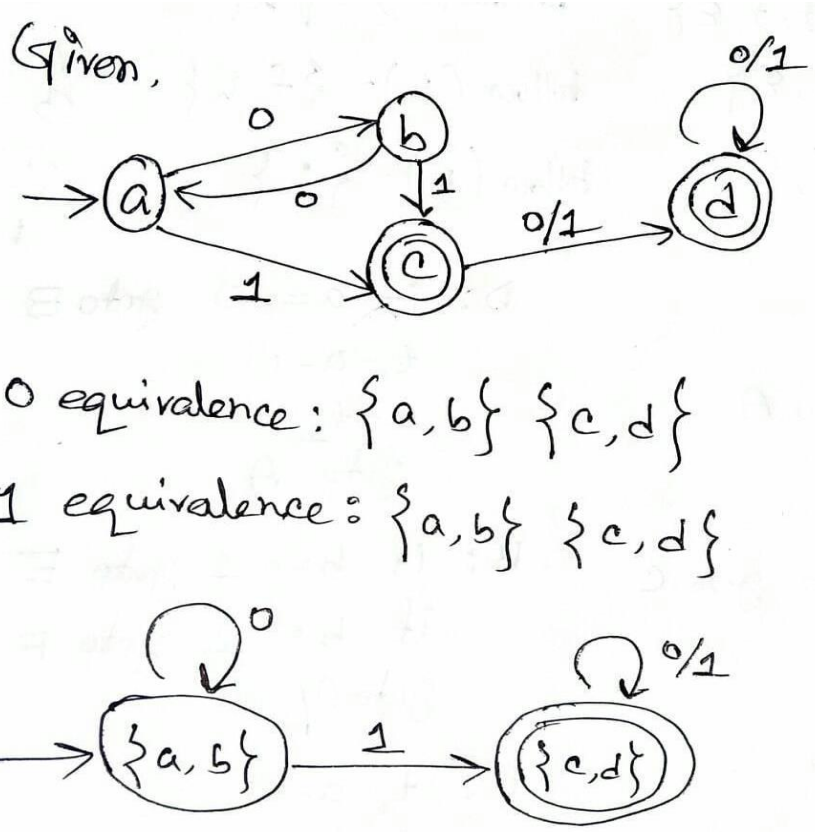
c) Considering the following examples design a CFG for C programming language-style function prototypes where Σ = {**void**, **int**, **float**, **id**, (,), ,, ;} **[5]**

Examples:

- void id (void);
- void id (int id);
- int id (int id, float id);
- float id (float id, float id, int id)

5. a.

b. **Solved by Younus-131**



c.

Question 6. [Marks: 14]

a) Illustrate the purpose of Left Factoring and the role of lookahead symbol in predictive parsing. **[4]**

b) Considering the following grammar develop an LL(1) parsing table and derive **(a)** using it. **[5]**

$$S \rightarrow (L) \mid a$$
$$L \rightarrow SX$$
$$X \rightarrow)SX \mid \varepsilon$$

c) Construct a DFA for the language of all strings consisting of **0**s and **1**s where the substring "10" comes in even numbers (>= 2). **[5]**
Example: **1010**, **10110**, (0100110, …

6. a.

b. **Solved by Younus-131**

Question 6. [Marks: 14]

a) Illustrate the purpose of Left Factoring and the role of lookahead symbol in predictive parsing. **[4]**

b) Considering the following Grammar develop an LL(1) parsing table and derive **(a)** using it. **[5]**

$$S \rightarrow (L) \mid a$$
$$L \rightarrow SX$$
$$X \rightarrow)SX \mid \varepsilon$$

c) Construct a DFA for the language of all strings consisting of **0**s and **1**s where the substring "10" comes in even numbers (>= 2). **[5]**
Example: **1010**, **10110**, (0100**1**0, …

first(S) = { (, a }
first(L) = { (, a }
first(X) = {), ε }

follow(S) = { $,) }
follow(L) = {) }
follow(X) = {) }

LL(1) parsing table:

	(	a	)	$
S	S→(L)	S→a		
L	L→SX	L→SX		
X			X→)SX X→ε	

It is not LL(1) Grammar.

(a) can't be derived because ")" contains two productions which are errors.

c. **Solved by Younus-131**

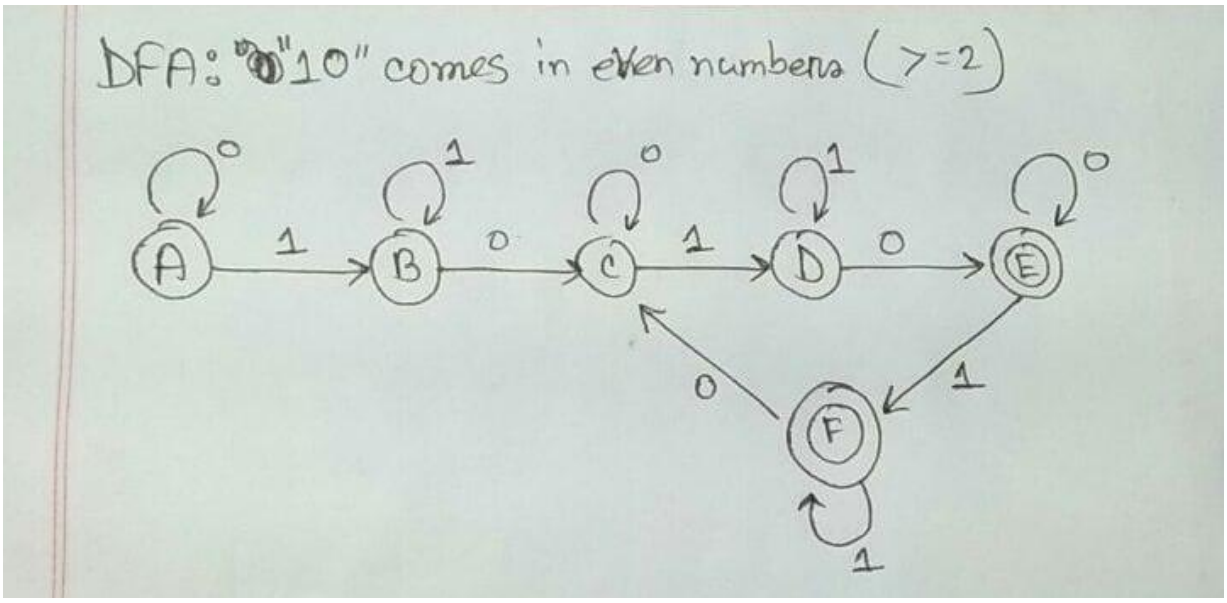
Question 6. [Marks: 14]

a) Illustrate the purpose of Left Factoring and the role of lookahead symbol in predictive parsing. **[4]**

b) Considering the following grammar develop an LL(1) parsing table and derive **(a)** using it. **[5]**

$$S \rightarrow (L) \mid a$$
$$L \rightarrow SX$$
$$X \rightarrow)SX \mid \varepsilon$$

c) Construct a DFA for the language of all strings consisting of **0**s and **1**s where the substring "10" comes in even numbers (>= 2). **[5]**
Example: **1010**, **10110**, (0100)30, …



Question 7. [Marks: 14]

a) Explain the concept of Syntax Directed translation and Attribute Grammar. **[4]**

b) Construct the LR(0) automaton for the following grammar: **[6]**

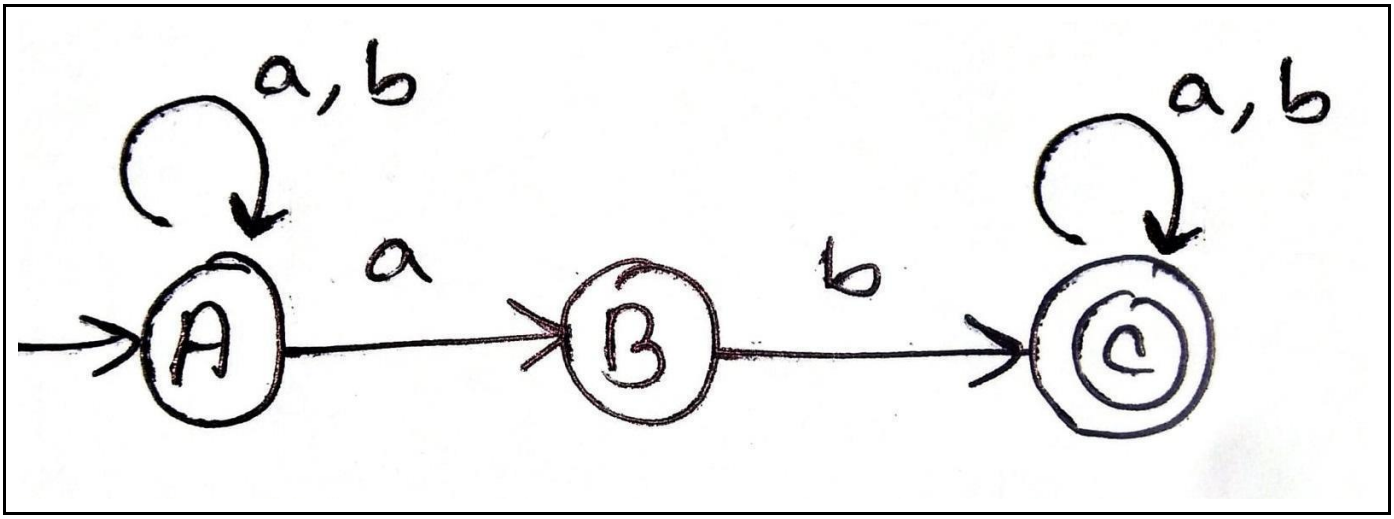
$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid id$$

c) Design an NFA accepting a set of strings over {a, b} in which each string of the language contains "ab" as substring. **[4]**

7.a.

b.

c. **Solved by Younus-131**



Quiz Questions

Quiz-1 Set-A

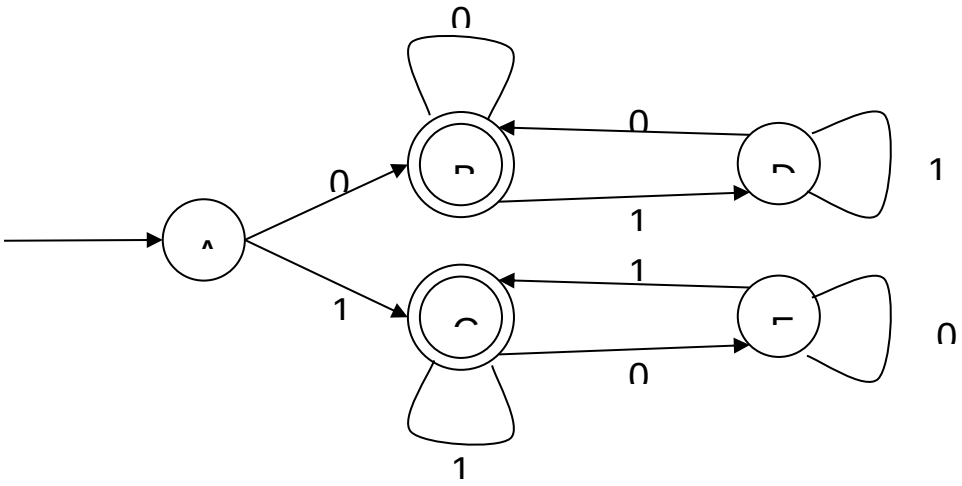
1. Explain the "Power of an Alphabet" with an example. [3]
2. Propose Regular Expression and construct a DFA for the language of all strings consisting of **0**s and **1**s which start and end with same symbol. [2+5]

Ans 1. *by 24*

Powers of an Alphabet:

- If Σ is an alphabet, the set of all strings of a certain length from that alphabet can be expressed by using an exponential notation.
- Σ^k is defined as the set of strings of length k, each of whose symbols is in Σ.
- $\Sigma^0 = \varepsilon$, no matter what the alphabet Σ is. In other words, ε is the only string of length 0.
- Example: If Σ = {a, b, c} then Σ^1 = {a, b, c}, Σ^2 = {aa, ab, ac, ba, bb, bc, ca, cb, cc}, Σ^3 = {aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc}.

Ans 2. **Solved by Younus-131**
Regular Expression: 0(0|1)*0 | 1(0|1)*1 | 0 | 1

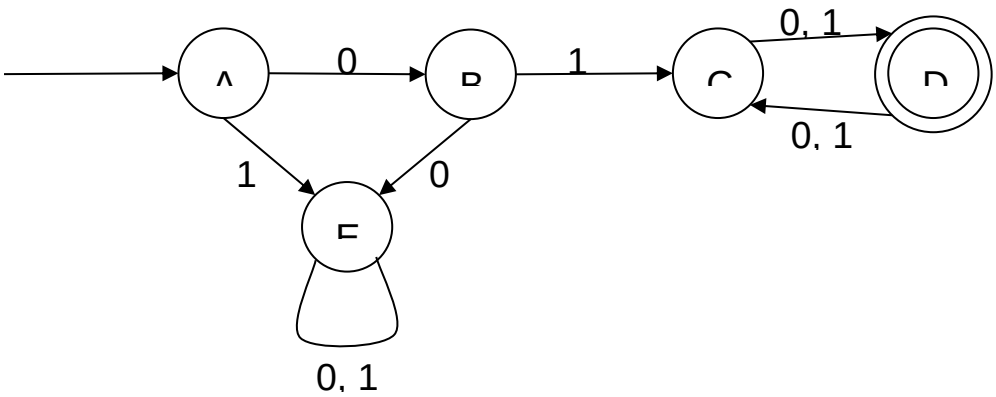


Quiz-1 Set-B

1. Explain the difference between DFA and NFA with the help of transition function. [3]
2. Propose Regular Expression and construct a DFA for the language of all strings consisting of **0**s and **1**s which are of ODD length and start with **01**. [2+5]

Ans 1. By 106
(Take from slide)
The difference between the DFA and the NFA is, for the NFA, small delta is a function that takes a state and input symbol as arguments (like the DFA's transition function), but returns a set of zero, one or more states (rather than exactly one state, as the DFA must.)

Ans 2. *by 24*
Regular Expression: 01 (0|1) ((0|1) (0|1))*



Quiz-2 Set-A

1. Explain the techniques to find the Generating and Reachable symbols of a CFG. [4]
2. Draw the transition diagram of the following DFA and minimize it. [6]

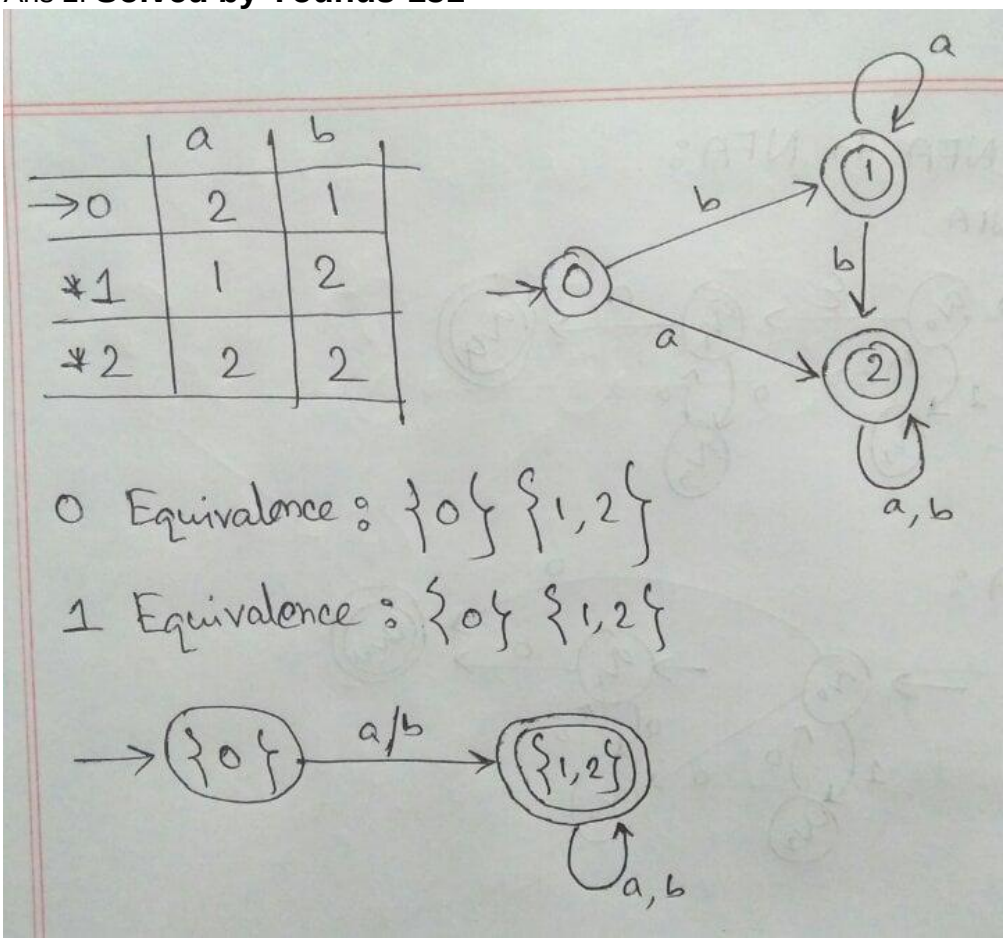
	a	b
→ 0	2	1
*1	1	2
*2	2	2

Ans 1. 109

- A symbol X is *generating* if $X \Rightarrow^* w$ for some terminal string w. Note that every terminal is generating, since w can be that terminal itself, which is derived by zero steps.
- A symbol X is *reachable* if there is a derivation $S \Rightarrow^* \alpha X \beta$ for some α and β.

- A symbol that is *useful* will be both *generating* and *reachable*.

Ans 2. **Solved by Younus-131**



Quiz-2 Set-B

1. Explain the importance of DFA Minimization and the conditions that need to be fulfilled to combine two states of a DFA into one. [4]

2. Eliminate useless symbols from the following grammar. [6]

$$S \rightarrow abS \mid abA \mid abB$$
$$A \rightarrow cd$$
$$B \rightarrow aB$$
$$C \rightarrow dc$$

Ans 1.

1(a)

- Minimization of DFA is required to obtain the minimal version of any DFA which consists of the minimum number of states possible.
- A DFA designed with 5 states and another DFA designed with 4 states, both doing the same task. Here both are correct but the DFA with 4 states is more efficient.

1(b)

- To minimize a DFA, we need to combine two states into one but it is possible when those two states are equivalent.

Two states are said to be equivalent if –

$\delta(A, X) \rightarrow F$ and $\delta(B, X) \rightarrow F$

OR where X is any input string.

$\delta(A, X) \not\rightarrow F$ and $\delta(B, X) \not\rightarrow F$

Ans 2. *by 24*

```
S -> abS | abA | abB
A -> cd
B -> aB
C -> dc
```

In the example above , production 'C -> dc' is useless because the variable 'C' will never occur in derivation of any string. The other productions are written in such a way that variable 'C' can never reached from the starting variable 'S'.

Production 'B ->aB' is also useless because there is no way it will ever terminate . If it never terminates , then it can never produce a string. Hence the production can never take part in any derivation.

To remove useless productions , we first find all the variables which will never lead to a terminal string such as variable 'B'. We then remove all the productions in which variable 'B' occurs.
So the modified grammar becomes –

```
S -> abS | abA
A -> cd
C -> dc
```

We then try to identify all the variables that can never be reached from the starting variable such as variable 'C'. We then remove all the productions in which variable 'C' occurs.
The grammar below is now free of useless productions –

```
S -> abS | abA
A -> cd
```

Quiz-3 Set-A

1. Explain two Error Recovery techniques. **[4]**

2. Do necessary changes in the following grammar with proper explanation to make it suitable for predictive parsing and find FIRST & FOLLOW sets for each non terminal of the modified grammar. **[2+4]**

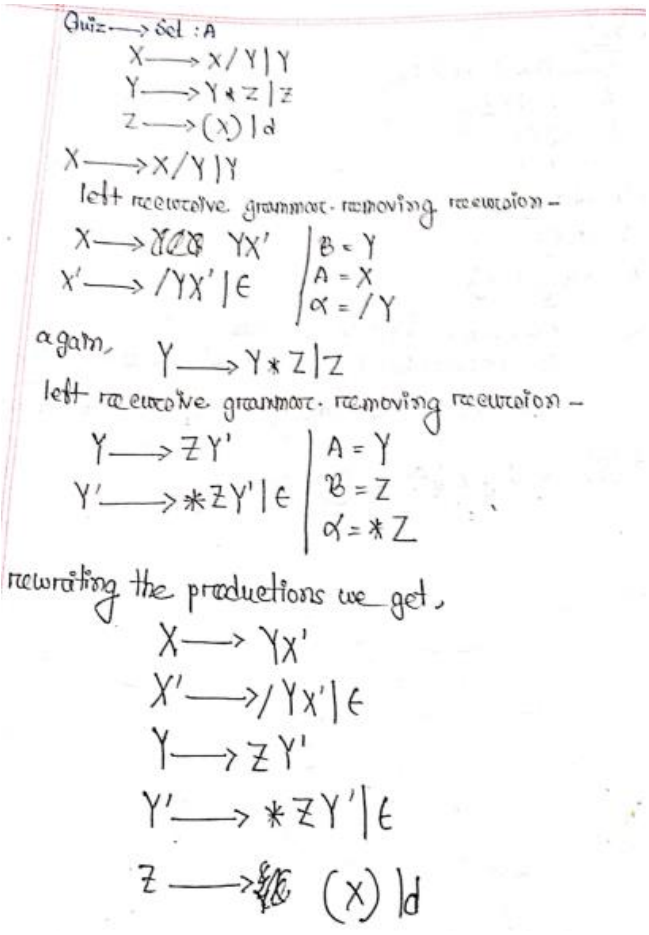
$$X \rightarrow X / Y \mid Y$$
$$Y \rightarrow Y * Z \mid Z$$
$$Z \rightarrow (X) \mid d$$

Ans 1. *by 24*

1. **Panic Mode:**
    - In this method, when an error is encountered anywhere in the statement, successive characters from the input are ignored one at a time until a designated set of synchronizing tokens is found. Synchronizing tokens are delimiters such as ; or } .
    - The advantage of this method is that it is the easiest to implement and guarantees not to go to infinite loop.
    - The disadvantage is that a considerable amount of input is skipped without checking it for additional errors.

2. **Statement Mode:**
    - In this method, when a parser encounters an error, it performs necessary correction on remaining input so that the rest of input statement allow the parser to parse ahead.
    - The correction can be deletion of extra semicolons, replacing comma by semicolon or inserting missing semicolon.
    - While performing correction, utmost care should be taken for not going in infinite loop.
    - Disadvantage is that it finds difficult to handle situations where actual error occurred before point of detection.

Ans 2. 0



First(Z) = { c, d }
First(Y') = { *, ϵ }
First(Y) = { c, d }
First(X') = { /, ϵ }
First(X) = { c, d }

6

Follow(X) = { $,) }
Follow(X') = { $,) }
Follow(Y) = { /, $,) }
Follow(Y') = { /, $,) }
Follow(Z) = { *, /, $,) }

Quiz-3 Set-B

1. Briefly explain how Predictive Parsing can avoid backtracking of Recursive-descent Parsing. **[4]**
2. Find FIRST & FOLLOW sets and construct the LL(1) parsing table for the following grammar. **[4+2]**

$$S \rightarrow AcBd$$
$$A \rightarrow bA \mid \epsilon$$
$$B \rightarrow cBd \mid a$$

Ans 1.

The problem with Recursive Descent Parsing is, if there are multiple productions for a non-terminal symbol, it may fail to choose the correct one, needing it to backtrack. We can prevent this backtracking by the following predictive parsing code:

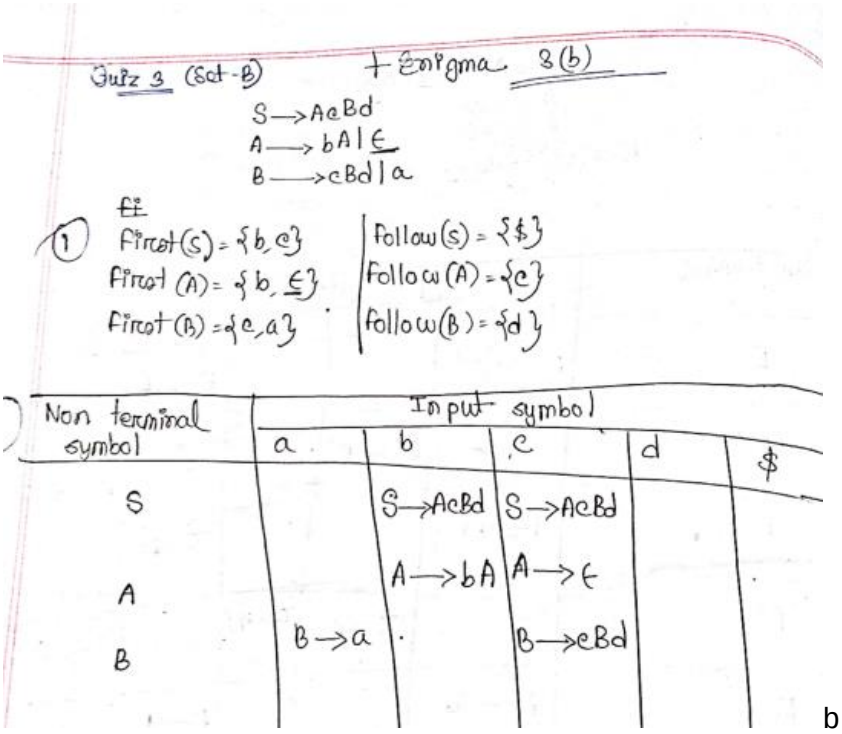
```
void stmt() {
        switch ( lookahead ) {
        case expr:
                match(expr); match(';'); break;
        case if:
                match(if); match('('); match(expr); match(')'); stmt();
                break;
        case for:
                match(for); match('(');
                optexpr(); match(';'); optexpr(); match(';'); optexpr();
                match(')'); stmt(); break;
        case other;
                match(other); break;
        default:
                report("syntax error");
        }
}

void optexpr() {
        if ( lookahead == expr ) match(expr);
}

void match(terminal t) {
        if ( lookahead == t ) lookahead = nextTerminal;
        else report("syntax error");
}
```

Figure 2.19: Pseudocode for a predictive parser

Ans 2.



Quiz 3 (Set-B) + Enigma 3(b)

S → AcBd
A → bA | ∈
B → cBd | a

ff

(1) First(S) = {b, c} | Follow(S) = {$}
First(A) = {b, ∈} | Follow(A) = {c}
First(B) = {c, a} | Follow(B) = {d}

Non terminal symbol	Input symbol a	b	c	d	$
S		S → AcBd	S → AcBd		
A		A → bA	A → ∈		
B	B → a		B → cBd		

b

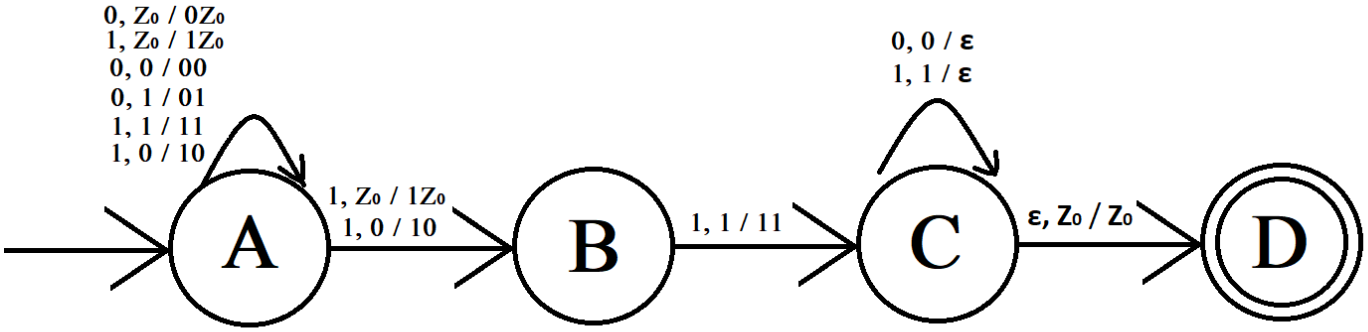
Assignment

1. Design PDAs for the following languages over the alphabet {0, 1}: [6]
   i. $L = \{w11w^R \mid w \text{ is in } (0+1)^*\}$
   ii. The language of all strings of 0's and 1's with an equal number of 0's and 1's.

2. Write down the Three Address Code for the following code snippet. [4]

```
a=4; b=2; cnt=0;
if(a<5)
    for(;;a--)
    {
      if(a==b)
      {
        b=a+b;
        cnt++;
        break;
      }
      cnt+=1;
    }
else
    switch(cnt)
    {
      case 1: func1(a, b);
      case 2: func1(cnt); break;
      case 3: a=3;
    }
```

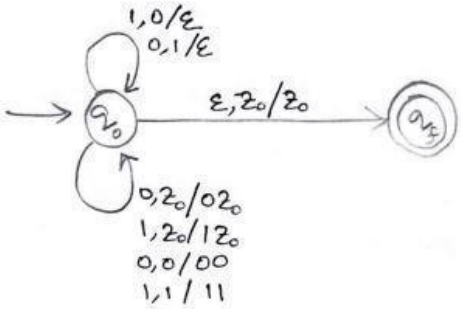
Ans 1.i. **Solved by Younus-131**



Ans 1.ii. **Solved by Younus-131**

ii Design a PDA for the given language over the alphabet $\{0,1\}$:

→ The language of all strings of 0's & 1's with an equal number of 0's and 1's.



$Q = \{q_0, q_f\}$

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

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

$q_0 = \{q_0\}$ (Initial State)

$Z_0 = \{Z_0\}$

$F = \{q_f\}$ (final state)

δ (Transition function):

$q_0, 0, Z_0 \rightarrow q_0, 0Z_0$

$q_0, 1, Z_0 \rightarrow q_0, 1Z_0$

$q_0, 0, 0 \rightarrow q_0, 00$

$q_0, 1, 1 \rightarrow q_0, 11$

$q_0, 1, 0 \rightarrow q_0, \varepsilon$

$q_0, 0, 1 \rightarrow q_0, \varepsilon$

$q_0, \varepsilon, Z_0 \rightarrow q_f, Z_0$

Ans 2. **Solved by Younus-131**

Three Address Code :

```
a = 4
b = 2
cnt = 0
if a<5 goto A
  goto X
A: if a==b goto B
   t1 = cnt+1
   cnt = t1
   t1 = a-1
   a = t1
   goto A
B: t1 = a+b
   b = t1
   t1 = cnt+1
   cnt = t1
   goto Z
X: if cnt==1 goto M
   if cnt==2 goto P
   if cnt == 3 goto Q
   goto Z
M: Param a
   Param b
   Call func1, 2
P: Param cnt
   Call func1, 1
   goto Z
Q: a = 3
Z : End
```

Origin Quiz-1 Set-A

1. Explain the procedure to get the lexemes from a source code with an example. [3]

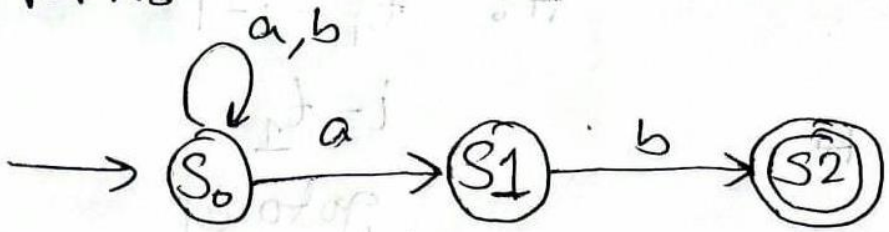
2. Convert the following NFA to DFA. [7]

State	a	b
→S0	{S0, S1}	{S0}
S1	∅	{S2}
*S2	∅	∅

Ans 1.

Ans 2. **Solved by Younus-131**
Ref: https://www.youtube.com/watch?v=LEigAZN6RdY

NFA:

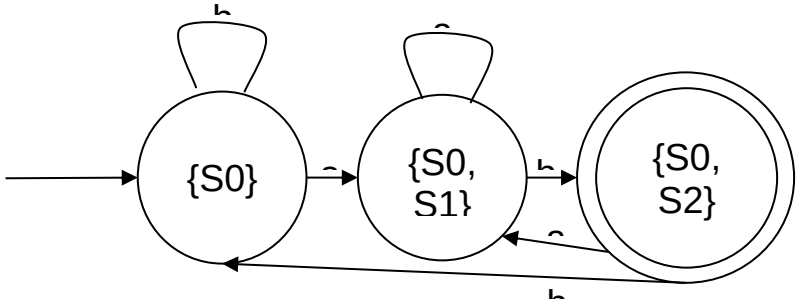


	a	b
→ S_0	$\{S_0, S_1\}$	$\{S_0\}$
S_1	$\emptyset$	$\{S_2\}$
*S_2	$\emptyset$	$\emptyset$

DFA:

	a	b
→ S_0	$\{S_0, S_1\}$	$\{S_0\}$
S_0S_1	$\{S_0, S_1\}$	$\{S_0, S_2\}$
*S_0S_2	$\{S_0, S_1\}$	$\{S_0\}$

DFA



Origin Quiz-1 Set-B

1. Explain the role of DFA in the procedure of tokenization. [3]

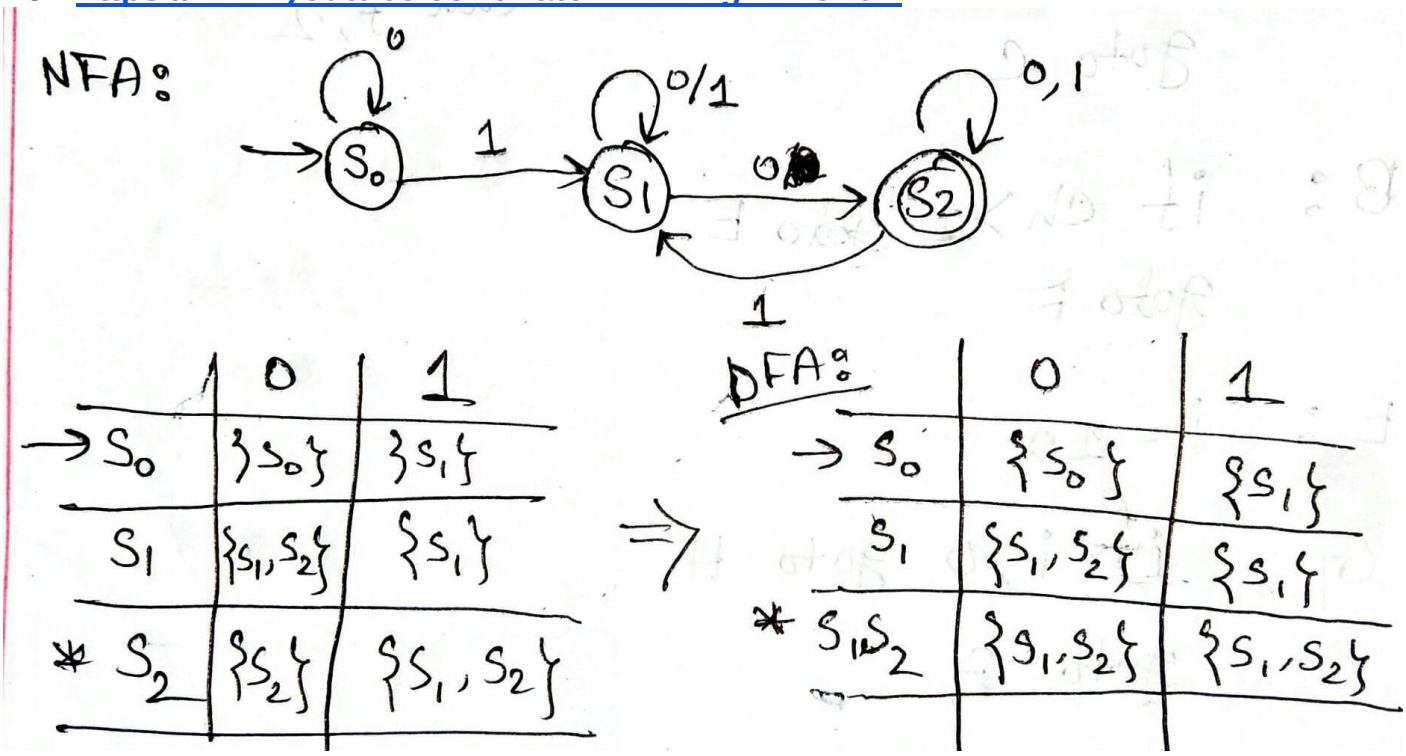
2. Convert the following NFA to DFA. [7]

State	0	1
→ **S0**	{S0}	{S1}
S1	{S1, S2}	{S1}
* **S2**	{S2}	{S1, S2}

Ans 1.

Ans 2. **Solved by Younus-131**

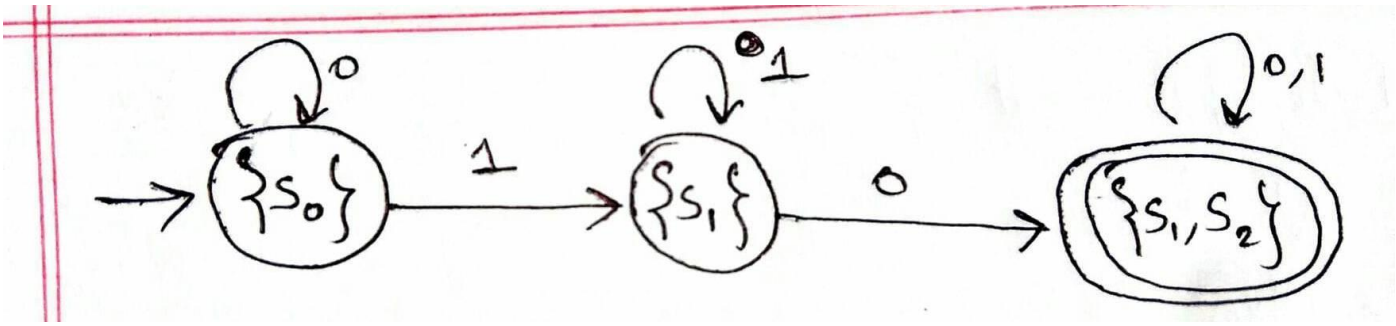
Ref: https://www.youtube.com/watch?v=LEigAZN6RdY



	0	1
→ S_0	$\{S_0\}$	$\{S_1\}$
S_1	$\{S_1, S_2\}$	$\{S_1\}$
* S_2	$\{S_2\}$	$\{S_1, S_2\}$

DFA:	0	1
→ S_0	$\{S_0\}$	$\{S_1\}$
S_1	$\{S_1, S_2\}$	$\{S_1\}$
* S_1S_2	$\{S_1, S_2\}$	$\{S_1, S_2\}$

DFA:



Origin Quiz-2 Set-A

1. What does the start symbol mean for a CFG? Explain with an example. [3]

2. Eliminate useless symbols from the following grammar. [7]

S --> AB / AC

A --> aAb / bAa / a

B --> bbA / aaB / AB

C --> abCA / aDb

D --> bD / aC

Ans 1. **Solved by Sujon-49**

A context-free grammar (CFG) is a formal system that consists of a set of production rules that specify how strings of symbols can be rewritten or derived into other strings of symbols.

The start symbol is a special non-terminal symbol that is used to indicate the beginning of a derivation or the starting point of a string in the language generated by the grammar. In other words, the start symbol is the symbol that appears on the left-hand side of the first production rule used to derive a string in the language.

For example, consider the following CFG:

S → aSb

S → ε

Here, S is the start symbol, and the two production rules say that S can be rewritten as "aSb" or as the empty string ε (which represents the string with no symbols). This grammar generates the language of all strings that can be formed by starting with any number of "a"s, followed by the same number of "b"s.

S → aSb

→ aaSbb

→ aab

→ aabbb

Ans 2.
By Id-19

S --> AB / AC

A --> aAb / bAa / a

B --> bbA / aaB / AB

C --> abCA / aDb

D --> bD / aC

In the example above, production D → bD | aC is useless because it can produce only more D ‘s and C’s, which can also only produce more D’s and C’s. Therefore, we can remove all productions involving D and remove D from the list of no-terminals.
And there is same rules for C → abCA | aDb, that’s why we can remove all productions involving C.

So, the useless symbols free grammar is:
S → AB
A → aAb | bAa | a
B → bbA | aaB | AB

Origin Quiz-2 Set-B

1. Define ambiguous grammar with an example. [3]

2. Eliminate useless symbols from the following grammar. [7]

A --> xyz / Xyzz

X --> Xz / xYz

Y --> yYy / Xz

Z --> Zy / z

Ans 1.

Id- 19:

An ambiguous grammar is a grammar that can produce multiple parse trees or interpretations for a single

input string. This can lead to ambiguity and difficulty in determining the intended meaning of the input.

Let,

S → aSb | bSa | ε

This grammar is ambiguous because some strings can have multiple parse trees. For example, the string **aabbaabb** can be generated in two ways:

S → aSb → aaSbb → aabbaabb

S → aSb → aSbb → aaSbbb → aabbaabb

Ans 2.
[Toufique]

A symbol is useful if it is both generating and reachable.
To eliminate useless symbols, we have to eliminate non-generating and non-reachable symbols.

Given,

A ➡ xyz / Xyzz

X ➡ Xz / xYz

Y ➡ yYy / Xz

Z ➡ Zy / z

There isn't any derivation where X =>* w or Y =>* w [where w is a terminal string]
So X and Y is not generating.
After eliminating X and Y from the productions we get,

A ➡ xyz

Z ➡ Zy / z

There isn't any derivation from start symbol A where A =>* **α**Z**β**
So Z is not reachable.
After eliminating Z from the productions we get,

A ➡ xyz

Origin Quiz-3 Set-A

1. Explain Left Factoring with an example. [3]

2. Find FIRST and FOLLOW set for the following grammar. [7]

$$S \rightarrow aBDh$$
$$B \rightarrow cC$$
$$C \rightarrow bC \mid \in$$
$$D \rightarrow EF$$
$$E \rightarrow g \mid \in$$
$$F \rightarrow f \mid \in$$

Ans 1.
Id - 19

Left factoring is a technique used in grammar to eliminate common prefixes of two or more production rules. It involves creating a new non-terminal symbol that represents the common prefix, making the grammar simpler and more efficient to parse.

Let,

S → ABc | ABd

A → a | ε

B → b | ε

After applying left factoring in those grammar,

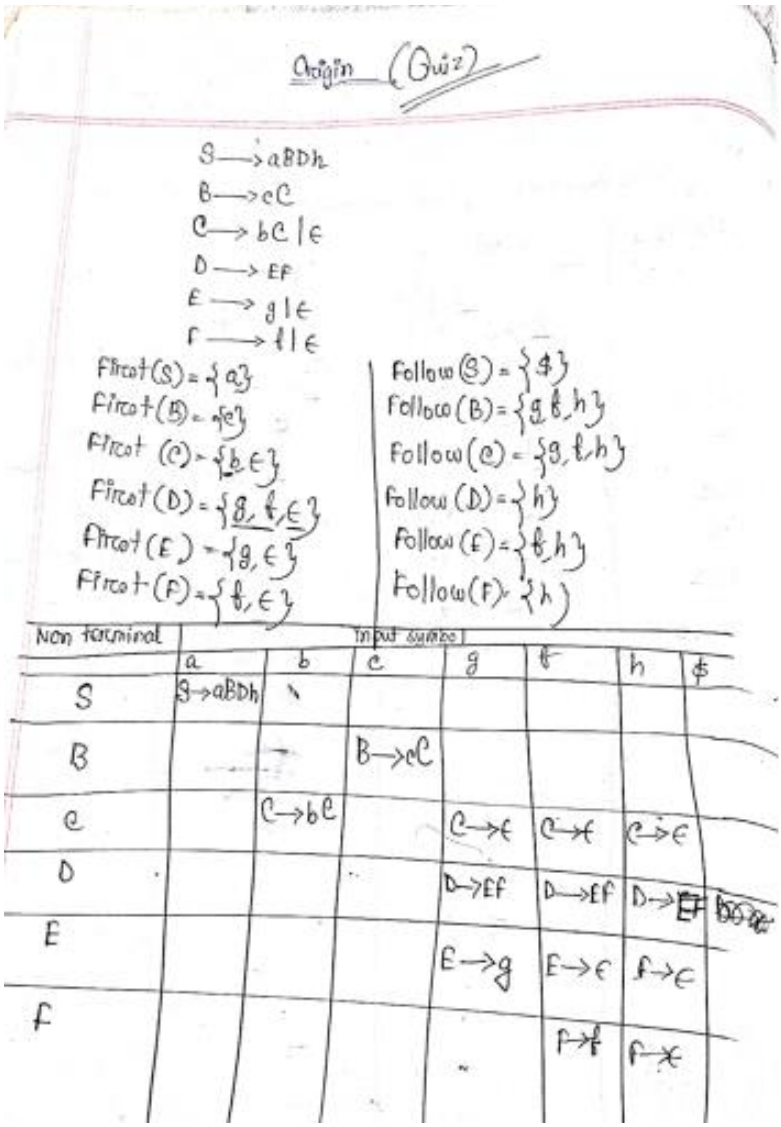
S → ABX

X → c | d

A → a | ε

B → b | ε

Ans 2.solved by 🐸



Origin Quiz-3 Set-B

1. Explain Left Recursion with an example. [3]

2. Find FIRST and FOLLOW set for the following grammar. [7]

S → aBDh
B → cC
C → bC | ϵ
D → EF
E → g | ϵ
F → f | ϵ

Ans 1.

Id - 19

Left recursion is a situation in which a non-terminal symbol can be replaced by a production rule that begins with the same non-terminal symbol. This can cause infinite recursion and make the grammar ambiguous and difficult to parse.

Let,

$A \rightarrow Aa \mid b$

After applying left recursion in this grammar,

$A \rightarrow bA'$

$A' \rightarrow aA' \mid \varepsilon$

Ans 2.
Id- 19

FIRST(S) = {a}
FIRST(B) = {c}
FIRST(C) = {b, ε}
FIRST(D) = {g, f, ε}
FIRST(E) = {g, ε}
FIRST(F) = {f, ε}
FIRST(a) = {a}
FIRST(b) = {b}
FIRST(c) = {c}
FIRST(g) = {g}
FIRST(f) = {f}

FOLLOW(S) = {$}
FOLLOW(B) = {g,f, h}
FOLLOW(C) = {g,f, h}
FOLLOW(D) = {h}
FOLLOW(E) = {f,h}
FOLLOW(F) = {h}

Origin-42 Final

Question 1. [Marks: 14]

a) Illustrate the differences among Deterministic Finite Automata (DFA), Nondeterministic Finite Automata (NFA) and Pushdown Automata (PDA). [4]

b) Explain all the components of a Context Free Grammar (CFG) with an example and describe the importance of CFG in compiler design. [5]

c) "Top-down parsing technique cannot work on some classes of CFGs". Justify the statement with appropriate examples. [5]

a) **Solved by Younus-131*:***

DFA	NFA	PDA
It can't use empty string transitions.	It can use empty string transitions.	It uses empty string transitions to reach the final state.
It consists of 5 tuples.	It also consists of 5 tuples.	It consists of 7 tuples.
For every symbol of the alphabet, there is only one state transition in DFA.	For every symbol of the alphabet, there may be multiple state transitions in NFA.	For every symbol of the alphabet, there may also have multiple state transitions in PDA.
It doesn't have the capability to store long sequences of input alphabets.	It also doesn't have the capability to store long sequences of input alphabets.	It has a stack to store the input alphabets.

B)

A CFG is a quadruple (4-tuple), that is, a system which consists of 4 elements. We describe a CFG, G as follows:

G = (V, T, P, S), where

V - finite nonempty set of *variables* or *non-terminal symbols*; Each variable represents a language.

T - finite nonempty set of *terminal symbols*, $V \cap T = \Phi$;

P - finite nonempty set of *productions* or grammar *rules* of the form,

$A \rightarrow \alpha$, where

[head] $A \in V$,

[body] $\alpha \in (V \cup T)^*$ and

[production symbol] '$\rightarrow$' means 'could take the value' / 'can be replaced with';

S - One of the variables represents the language being defined; it is called the *start symbol*, $S \in V$.

Example:

G_1:

$S \rightarrow Abb$,
$A \rightarrow \varepsilon$
$A \rightarrow aA$
$A \rightarrow bA$.

OR

$S \rightarrow Abb$,
$A \rightarrow \varepsilon \mid aA \mid bA$.

According to the Formal Definition,

$G_1 = (\{S, A\}, \{a, b\}, \{S \rightarrow Abb, A \rightarrow \varepsilon, A \rightarrow aA, A \rightarrow bA\}, S)$.

c)

Question 2. [Marks: 14]

a) Draw the transition diagram of the following NFA and construct a DFA equivalent to it. [6]

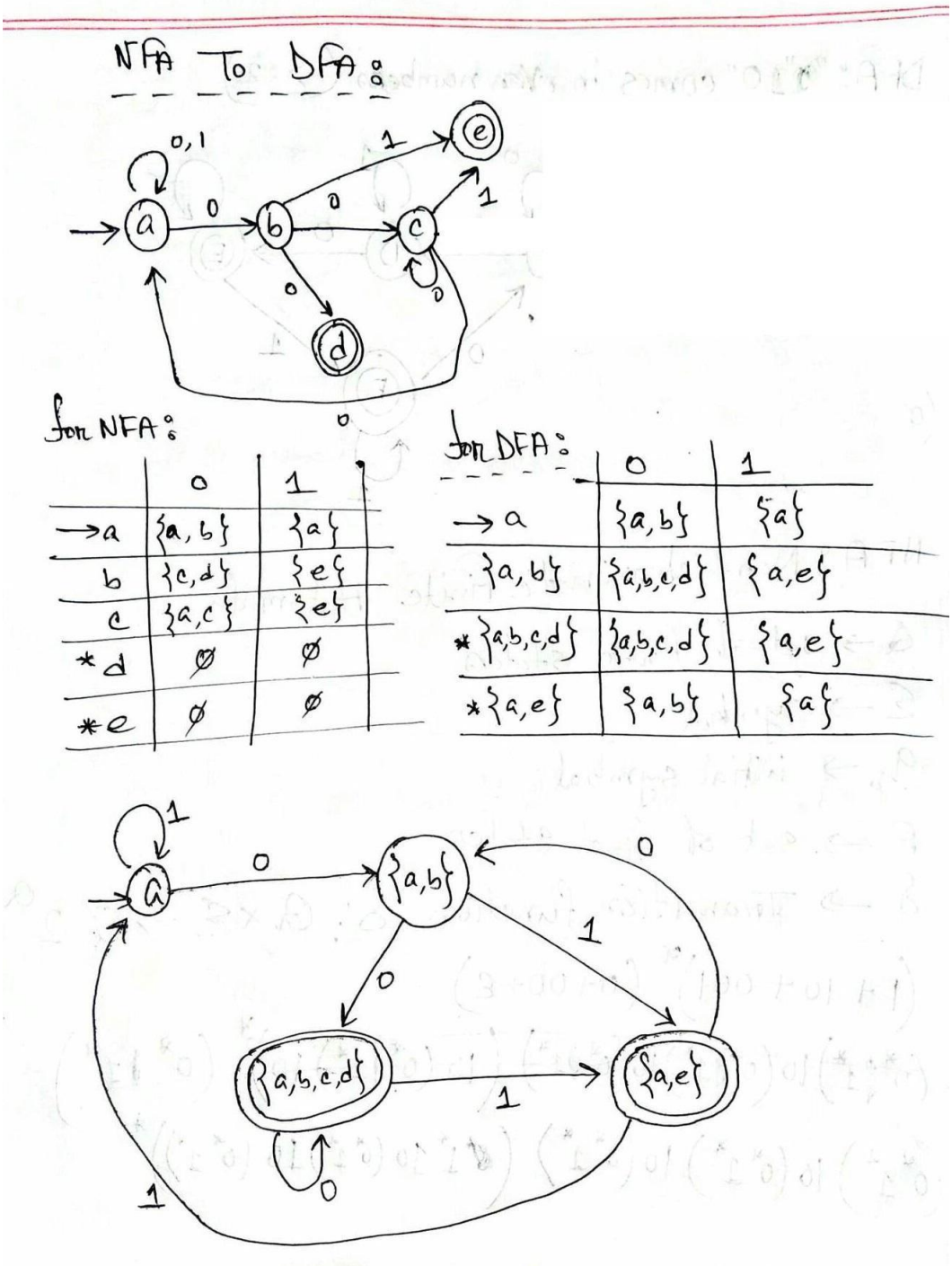
	0	1
→ a	{a, b}	{a}
b	{c, d}	{e}
c	{a, c}	{e}
*d	∅	∅
*e	∅	∅

b) Construct DFAs accepting the following strings over the alphabet {0, 1}: [8]

i. The set of all strings that do not contain three consecutive zeros.

ii. The set of all strings where the substring "10" comes in even numbers (>= 2).
[example: **1010, 10110**, 0100100, ...]

a) **Solved by Younus-131**
Ref: https://youtu.be/LEigAZN6RdY



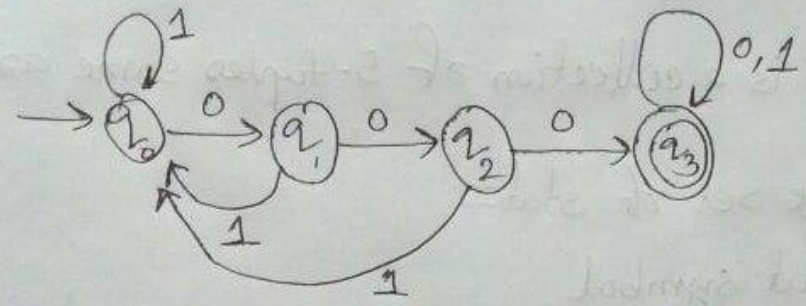
	0	1
→a	{a, b}	{a}
b	{c, d}	{e}
c	{a, c}	{e}
*d	∅	∅
*e	∅	∅

	0	1
→a	{a, b}	{a}
{a, b}	{a, b, c, d}	{a, e}
*{a, b, c, d}	{a, b, c, d}	{a, e}
*{a, e}	{a, b}	{a}

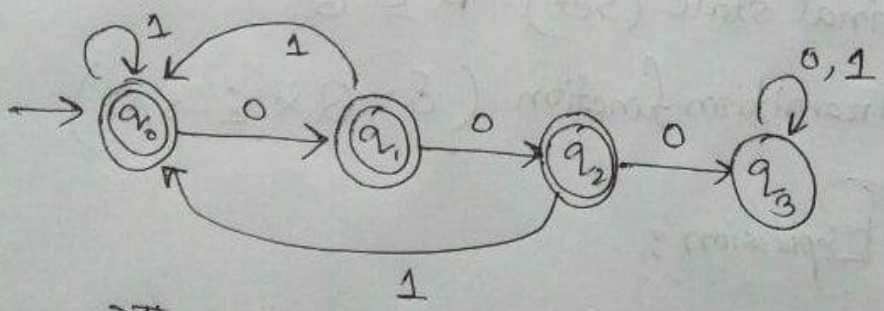
Final state 👍:{abcd} & {ae}

b.i. **Solved by Younus-131**

→ DFA with 3 consecutive 0s



→ DFA without 3 consecutive 0s

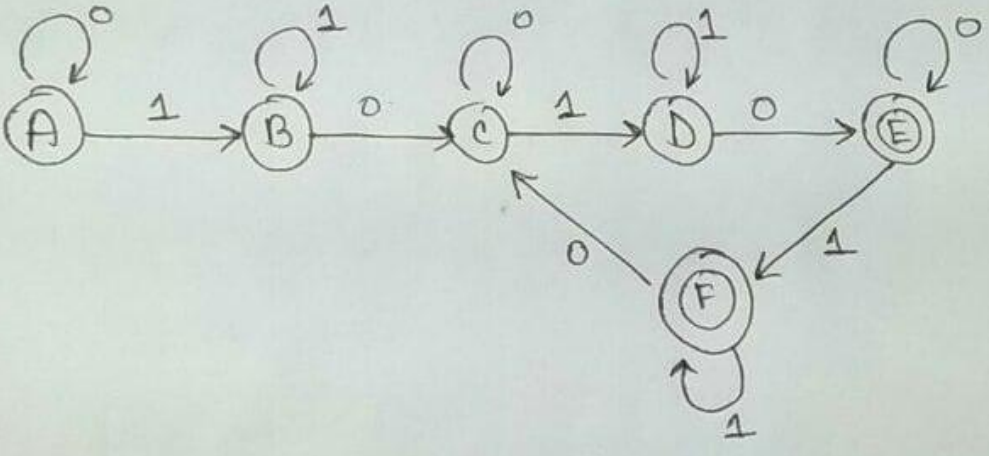


১ম

উপরের DFA এর Non-final কে final & final state কে Non-final করা হয়েছে।

b.ii. **Solved by Younus-131**

DFA: "10" comes in even numbers (>=2)



Question 3. [Marks: 14]

a) Propose Regular Expressions for the following languages: [6]

i. L = {w | w consists of As and Bs in which two Bs do not come together.}

ii. L = {w | w consists of 0s and 1s which is of ODD length and starts with 10}

iii. L = {2, 12, 112, 200, 1200, 11120000, …}

iv. L = {0111, 01011, 01111, 010011, 010111, 011011, 011111, 0100011, …}

Ans: **Solved by Younus-131**

a) i. (A + BA)* (B + Ɛ) - **Explanation :** https://youtube.com/watch?v=IswH4GqO8Sg&feature=share

alternative : (B + Ɛ) (A + AB)* ***Both are correct***

ii. 10 (0|1) ((0|1) (0|1))*

iii. 1* 2 0*

iv. 01 (0|1)* 11

3.b.

b) Consider the grammar and answer following questions:

S → iEtS | iEtSeS | a

E → b

i. Left-factor the grammar. [1]

ii. Find FIRST and FOLLOW sets. [3]

iii. Construct the LL(1) parsing table. [2]

iv. Is the modified grammar suitable for top-down parsing? Justify your answer based on the LL(1) table. [2]

Ans:

Tamal 122

i) S => iEtSS'|a
S' => epsilon | eS

ii) First(S) = {i, a}, First (S`) = {e, epsilon}, First(E) = {b}
Follow (S) = {$, e}, Follow(S`) = {$, e} Follow (E) = {t}

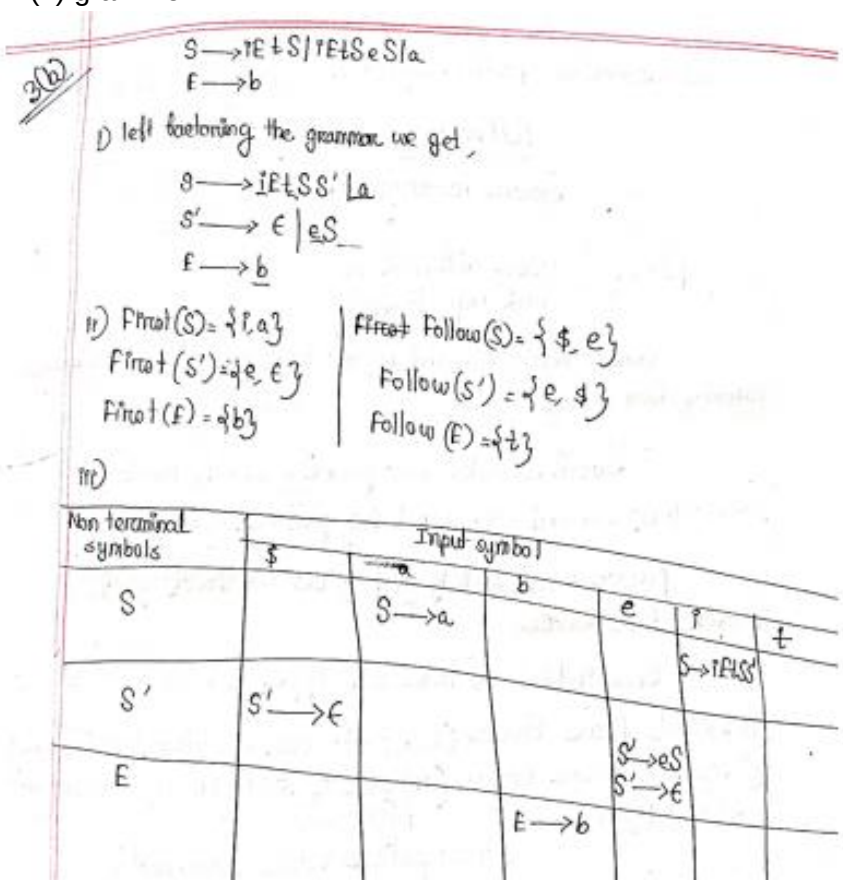
iii)

	i	t	e	a	b	$

S	1			2		
S`			3,4			4
E					5	

Not LL(1) grammer

iv) no suitable as not LL(1) grammer



Question 4. [Marks: 14]

Write down the Three Address Code for the following code snippet. [6]

```
i=1;
a=0;
for( ; i<5; i++)
    for(j=i; j<4; j++)
        a=a+1;
    if(i==3)
        break;
switch(i):
{
    case 0: a--;
            break;
    case 1: j=0;
    case 2: a=0;
}
```

a) Ans: **Solved by Younus-131**

i = 1
a = 0

A : if i < 5 goto B
goto F

B : j = i

C : if j < 4 goto D
goto E

D : $t_1 = a + 1$
$a = t_1$
$t_1 = j + 1$
$j = t_1$
goto C

E : if i == 3 goto F
$t_1 = i + 1$
$i = t_1$
goto A

F : if i == 0 goto G
if i == 1 goto H
if i == 2 goto I
goto Z

G : $t_1 = a - 1$
$a = t_1$
goto Z

H : j = 0

I : a = 0

Z : End

4.b.

Consider the grammar and answer following questions:

$S \rightarrow 2X \mid 1Y \mid Z$
$X \rightarrow 2S \mid 22$
$Y \rightarrow 1 \mid X$
$Z \rightarrow 21$

i. Remove unit productions from the grammar. [5]
ii. Eliminate useless symbols from the modified grammar. [3]

Ans: *by 24*

i

S → 2X | 1Y | Z
X → 2S | 22
Y → 1 | X
Z → 21

So,

(S,S), (S,Z)
(X,X)
(Y,Y), (Y,X)
(Z,Z)

Pair	Production
(S,S)	S → 2X | 1Y
(S,Z)	S → 21
(X,X)	X → 2S | 22
(Y,Y)	Y → 1
(Z,Z)	Z → 21
(Y,X)	Y → 2S | 22

After eliminating unit production

S → 2X | 1Y | 21
X → 2S | 22
Y → 1 | 2S | 22
Z → 21

ii

Modified grammars

$S \rightarrow 2X \mid 1Y \mid 21$

$X \rightarrow 2S \mid 22$

$Y \rightarrow 2S \mid 22 \mid 1$

$Z \rightarrow 21$

Here, productions,

$S \rightarrow 2X$, $S \rightarrow 1Y$, $S \rightarrow 21$

$X \rightarrow 2S$, $X \rightarrow 22$

$Y \rightarrow 2S$, $Y \rightarrow 22$, $Y \rightarrow 1$

$Z \rightarrow 21$

Generating

Terminals : (21, 22, 1)

Head of ~~ter~~ generating : (S, X, Y, Z)

∴ Non-generating : ϕ

Reachable

Variable that can be reached from start symbol.

∴ ~~X~~ Unreacheable : $Z \rightarrow 21$

∴ After eliminating useless symbols:

$S \rightarrow 2X \mid 1Y \mid 21$

$X \rightarrow 2S \mid 22$

$Y \rightarrow 2S \mid 22 \mid 1$

Question 5. [Marks: 14]

a) How can the following grammar be made suitable for top-down parsing? [4]

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

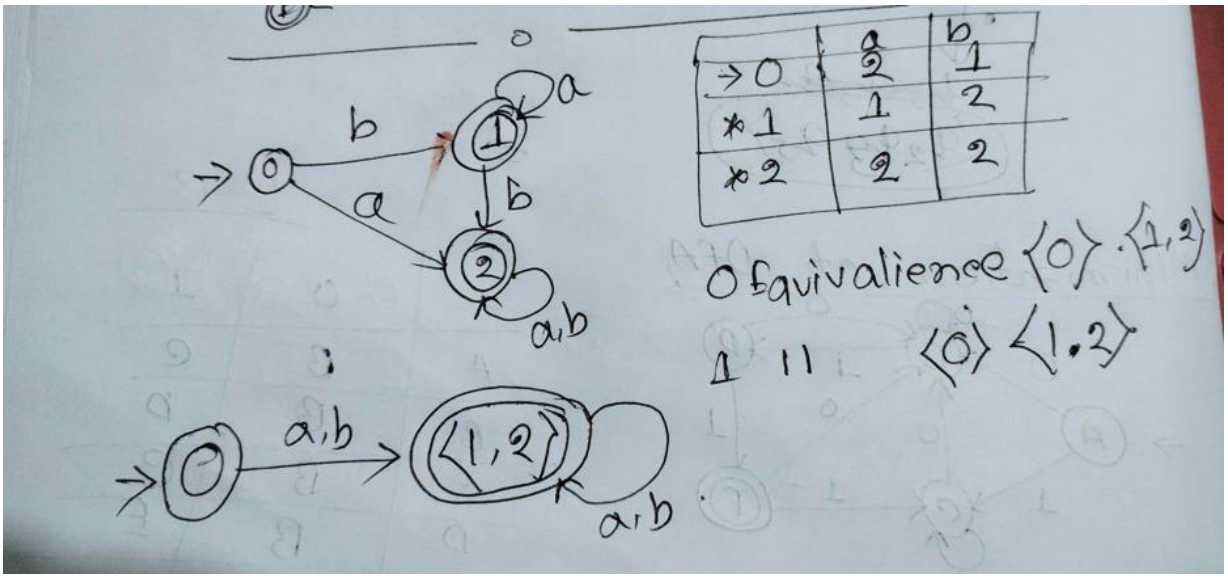
Ans:
Tamal 122
S -> (L) | a
L -> SL'

L' -> epsilon | ,SL'

b) Draw the transition diagram of the following DFA and minimize it. [5]

	a	b
→0	2	1
*1	1	2
*2	2	2

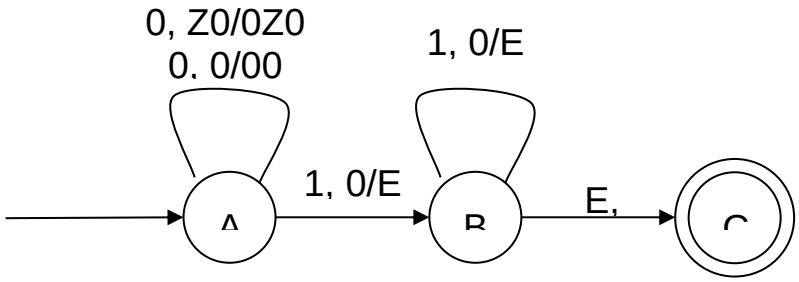
Ans:



.ç .

c) Construct a PDA that accepts $L = \{0^n 1^n \mid n \geq 1\}$ with proper explanation. [5]

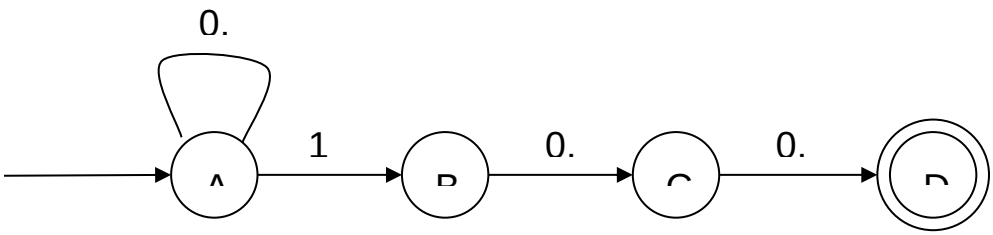
Ans: *by 24*



Question 6. [Marks: 14]

a) Design an NFA that accepts all strings over {0, 1} where the 3rd symbol from the end is a 1. [4]

Ans: *by 24*



b) Write short notes with appropriate examples on: [4]

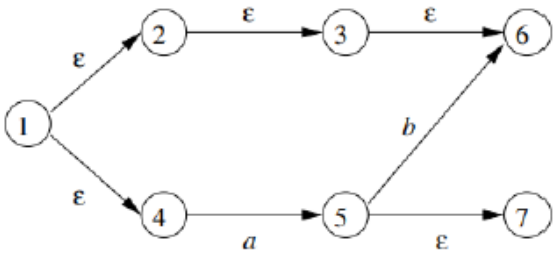
i. Epsilon-closure of a state of an ε-NFA

ii. Powers of an alphabet [6]

Ans: *by 24*

Epsilon-Closures (ε*):

All the states that can be reached from a particular state only by seeing the ε symbol.



ECLOSE (3) = {3, 6}

ECLOSE (2) = {2, 3, 6}

ECLOSE (1) = {1, 2, 4, 3, 6}

Powers of an Alphabet:

- If Σ is an alphabet, the set of all strings of a certain length from that alphabet can be expressed by using an exponential notation.
- Σ^k is defined as the set of strings of length k, each of whose symbols is in Σ.
- $\Sigma^0 = \varepsilon$, no matter what the alphabet Σ is. In other words, ε is the only string of length 0.
- Example: If Σ = {a, b, c} then Σ^1 = {a, b, c}, Σ^2 = {aa, ab, ac, ba, bb, bc, ca, cb, cc}, Σ^3 = {aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc}.

c) Briefly describe the error recovery techniques that can be applied during parsing and explain how an error recovery technique can develop infinite loop within the procedure itself. [6]

Ans: *by 24*

Question 7. [Marks: 14]

Consider the grammar and answer following questions:

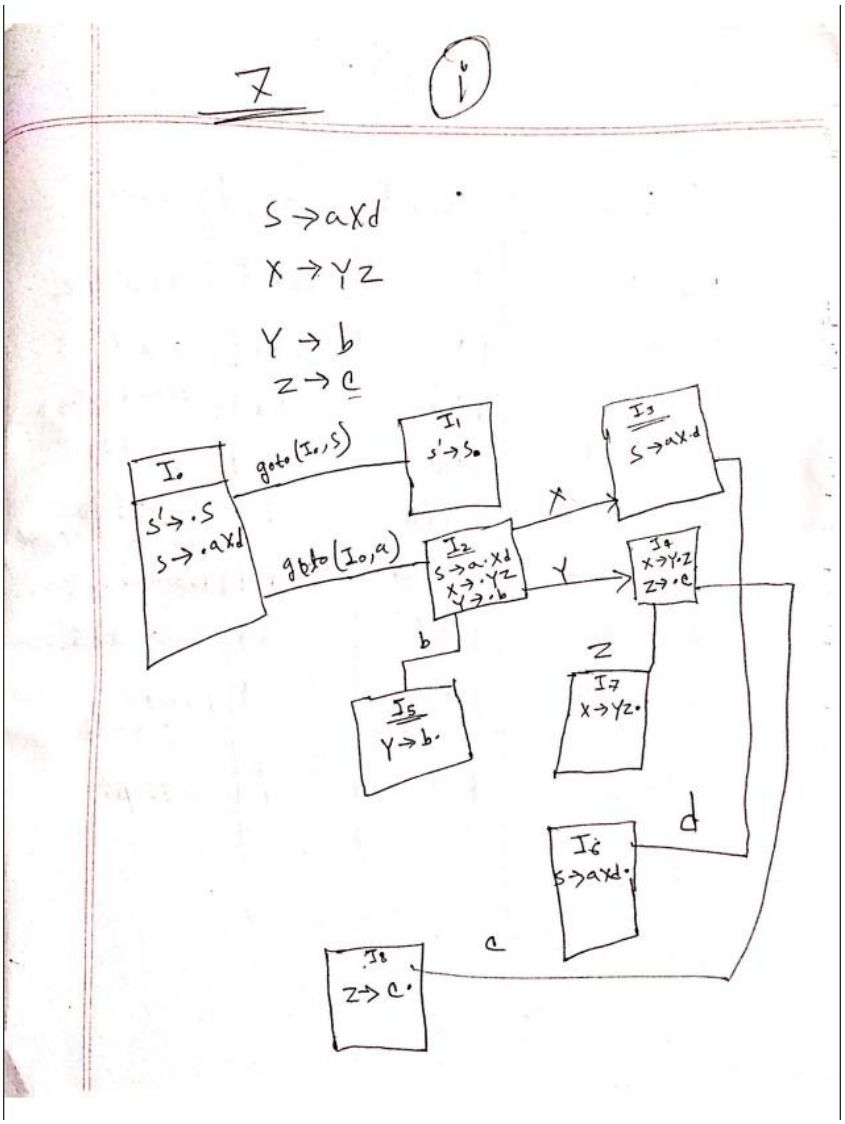
1. $S \rightarrow aXd$
2. $X \rightarrow YZ$
3. $Y \rightarrow b$
4. $Z \rightarrow c$

i. Construct the LR(0) automaton. [7]

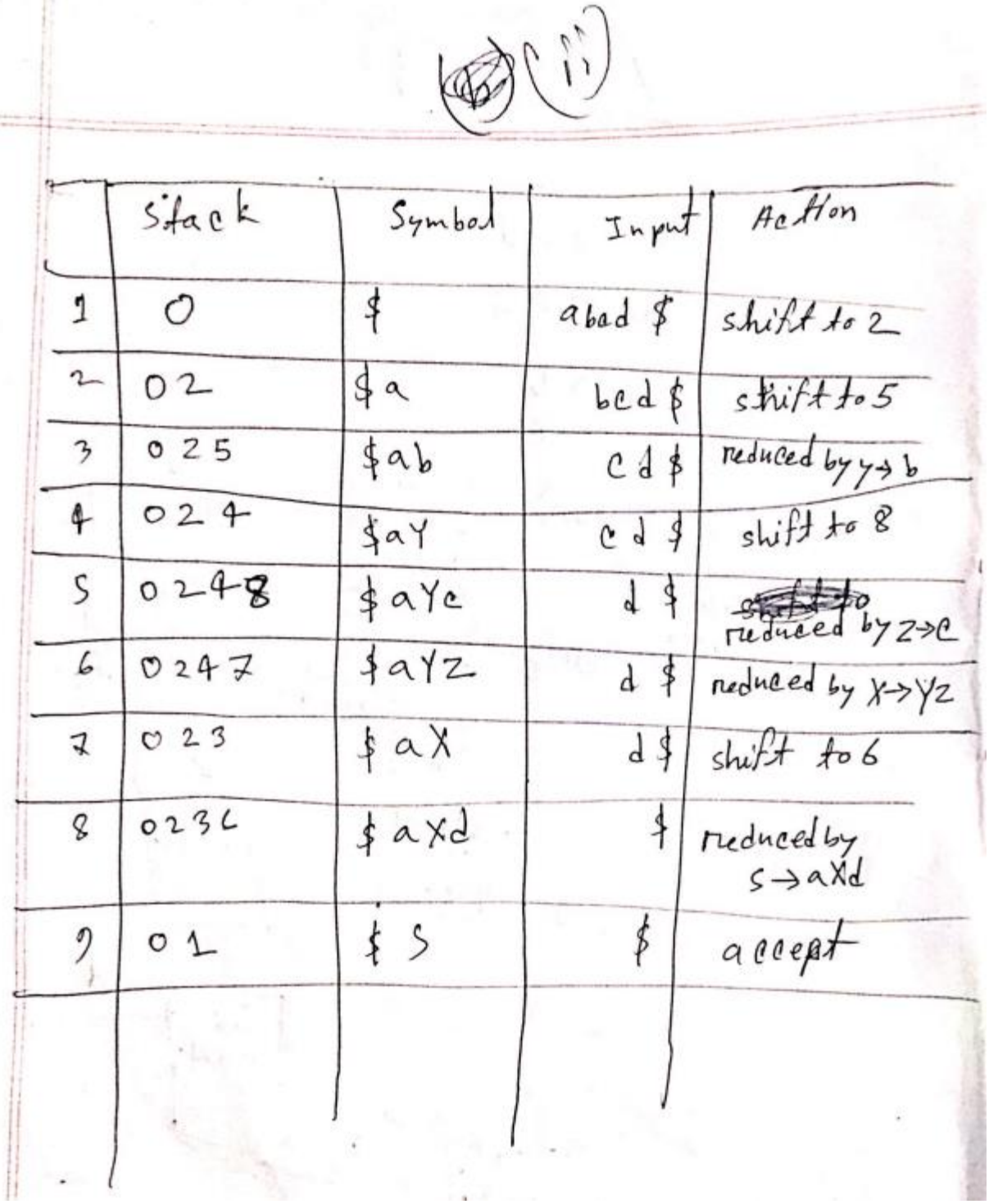
ii. Illustrate the actions of a shift-reduce parser on input **abcd** using the LR(0) automaton. [4]

iii. Construct an LR parsing table. [3]

7.i. **(Mukaffi - 77)**

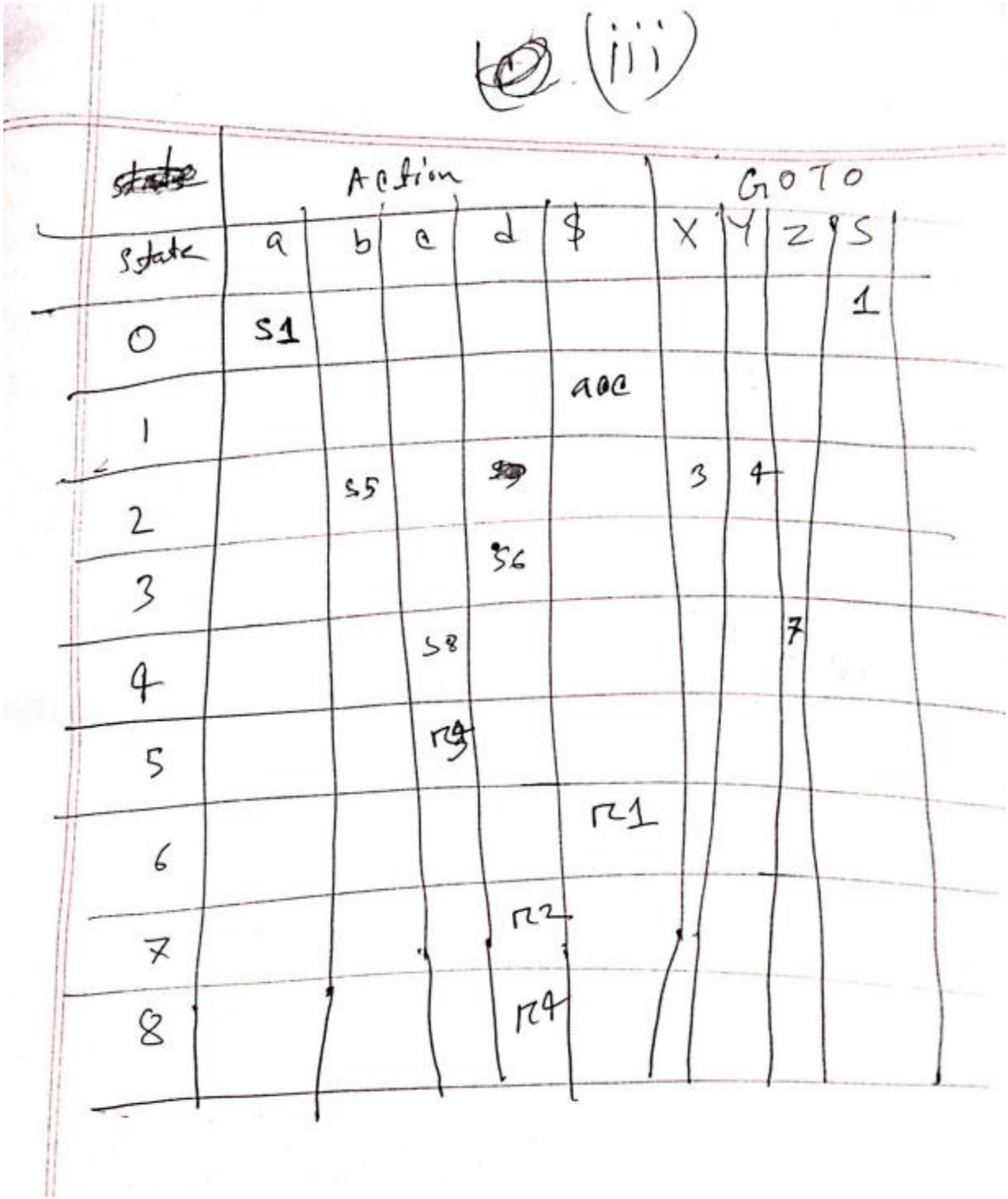


7.ii.



	Stack	Symbol	Input	Action
1	0	$	abad $	shift to 2
2	02	$a	bcd $	shift to 5
3	025	$ab	cd $	reduced by Y→b
4	024	$aY	cd $	shift to 8
5	0248	$aYc	d $	reduced by Z→c
6	0247	$aYZ	d $	reduced by X→YZ
7	023	$aX	d $	shift to 6
8	0236	$aXd	$	reduced by S→aXd
9	01	$S	$	accept

7.iii.



Enigma-41 Final

Question 1. [Marks: 14]

a) Define Context Free Grammar (CFG) describing all of its components. Why are these grammars called context free? [4]

b) What are the purposes of Linker and Loader? Write down the benefits of Intermediate Representation of a source program in a compiler. [4]

c) Define Compiler. Draw the block diagram of a compiler and briefly describe different components of it. [6]

a.

G_1:

$S \to Abb$,
$A \to \varepsilon$
$A \to aA$
$A \to bA$.

OR

$S \to Abb$,
$A \to \varepsilon \mid aA \mid bA$.

According to the Formal Definition,

$$G_1 = (\{S, A\}, \{a, b\}, \{S \to Abb, A \to \varepsilon, A \to aA, A \to bA\}, S).$$

- Derivation of Strings of terminals, for example, babb:
  $S \Rightarrow Abb \Rightarrow bAbb \Rightarrow baAbb \Rightarrow babb$.
  In short, $S \Rightarrow^* babb$. [From start symbol to the string]
- We can check in another way that this string can be derived with the given grammar:
  $babb \Rightarrow baAbb \Rightarrow bAbb \Rightarrow Abb \Rightarrow S$. [From the string to the start symbol; With these two types of derivation, we can relate top-down and bottom-up approaches.]

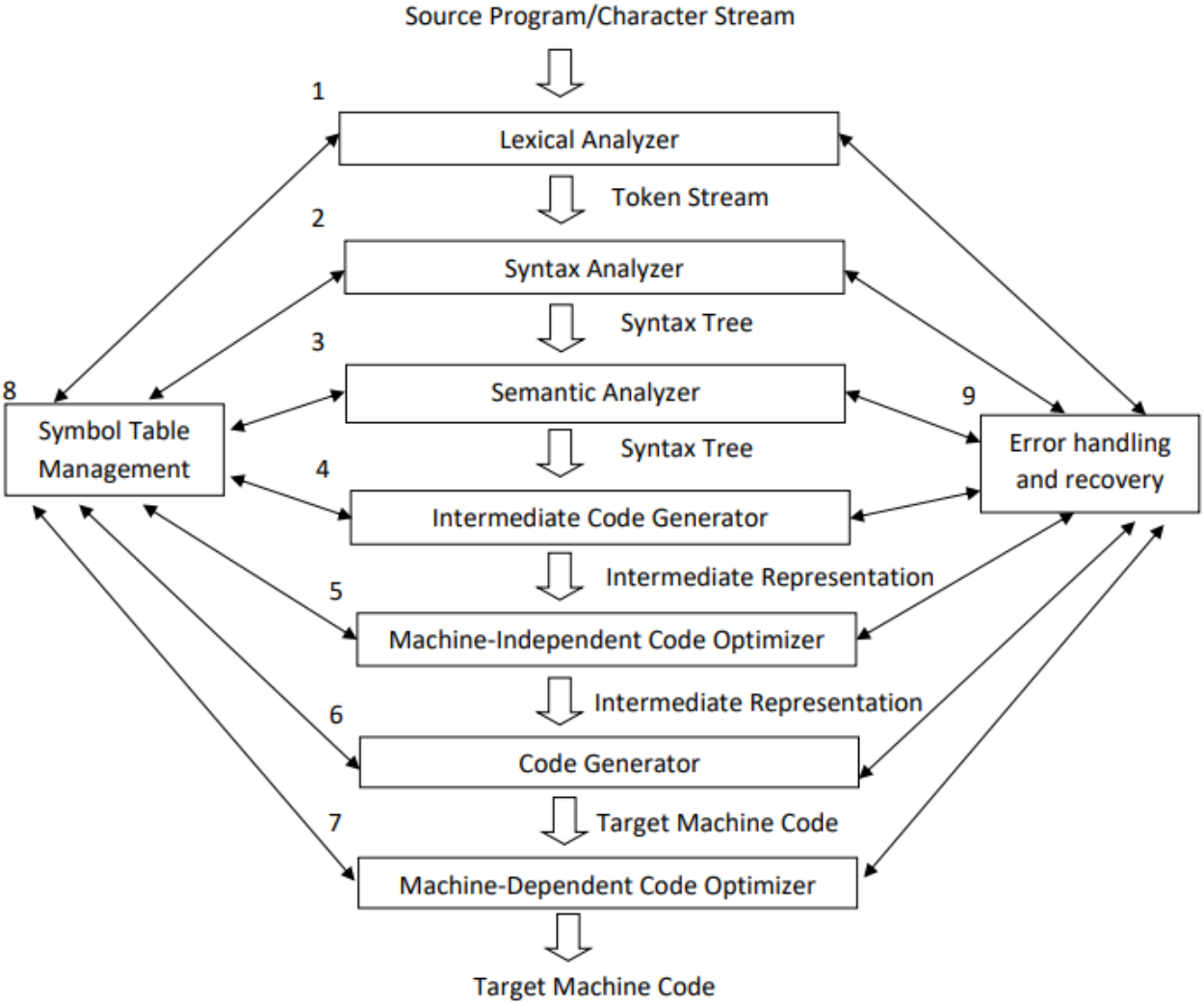
- We derive similarly, $S \Rightarrow^* bb$, $S \Rightarrow^* abb$, $S \Rightarrow^* bbb$, $S \Rightarrow^* aabb$, $S \Rightarrow^* abbb$, $S \Rightarrow^* babb$, $S \Rightarrow^* bbbb$, $S \Rightarrow^* aaabb$, …
- G_1 'generates' {bb, abb, bbb, aabb, abbb, babb, bbbb, aaabb, …}, we denote it by $L(G_1)$, which is said to be a *Context Free Language (CFL)*, as it is generated by a CFG.

In a production of a CFG, substitution of 'A' within '$\alpha_1 A \alpha_2$' does not depend on α_1 and α_2, that is, context α_1-α_2, and so, these types of grammars are called *context-free* grammars.

b.

c. Tamal 122

Structure of a Compiler



Major Phases of a Compiler with phase inputs and outputs

Compiler: A compiler is a program that can read a program in one language – the *source language* – and translate it into an equivalent program in another language – the *target language*.

Question 2. [Marks: 14]

a) Draw the transition diagram of the following ε-NFA and construct a DFA equivalent to it. [6]

	ε	a	b
→ *1	{3}	∅	{2}
2	∅	{2, 3}	{3}
3	∅	{1, 3}	∅

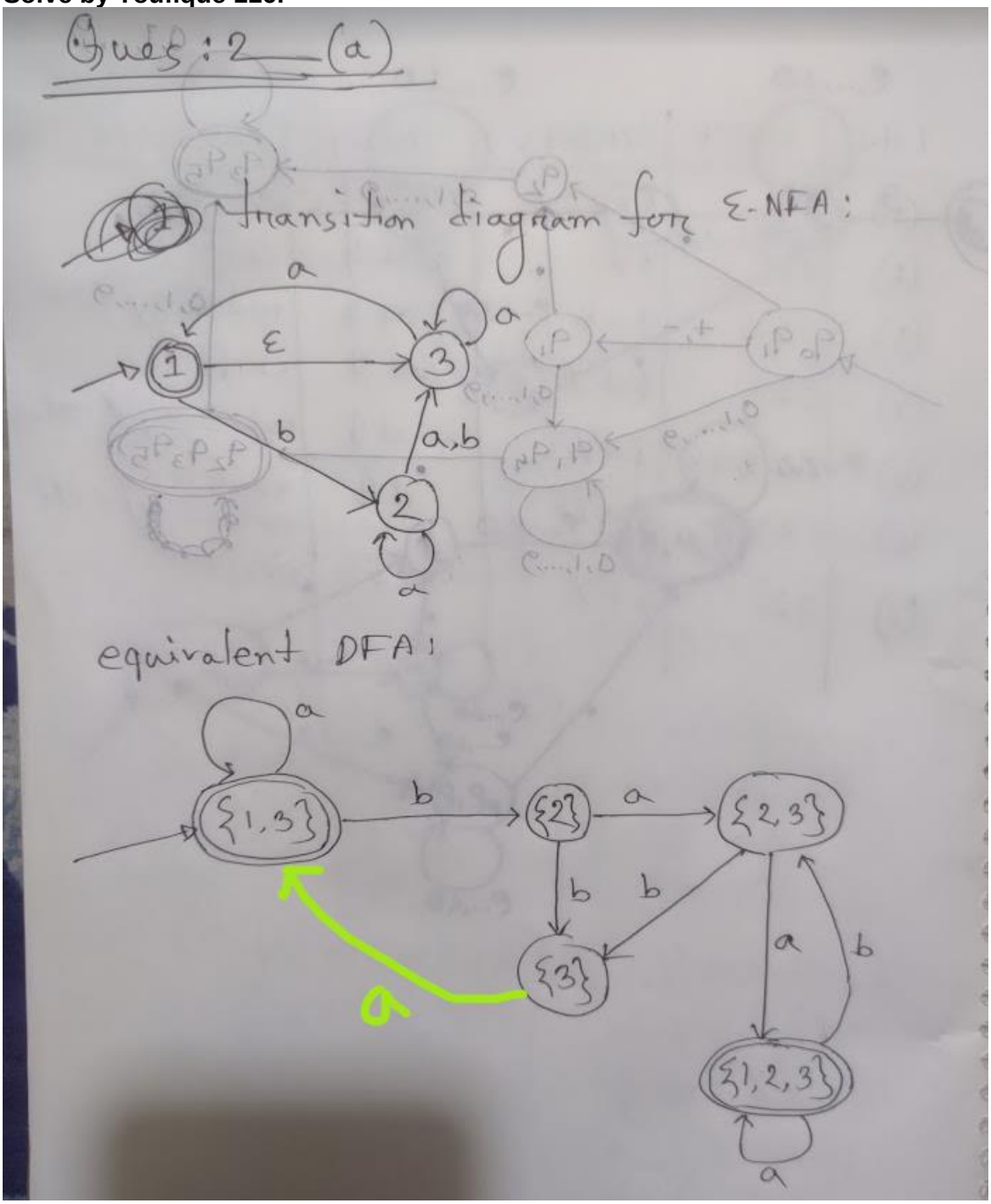
b) Design DFAs accepting the following strings over the alphabet {0, 1}: [8]

i. The set of all strings that contains a substring of "2 or more 0s followed by a 1".

ii. The set of all strings such that the number of 1 is multiple of 3.

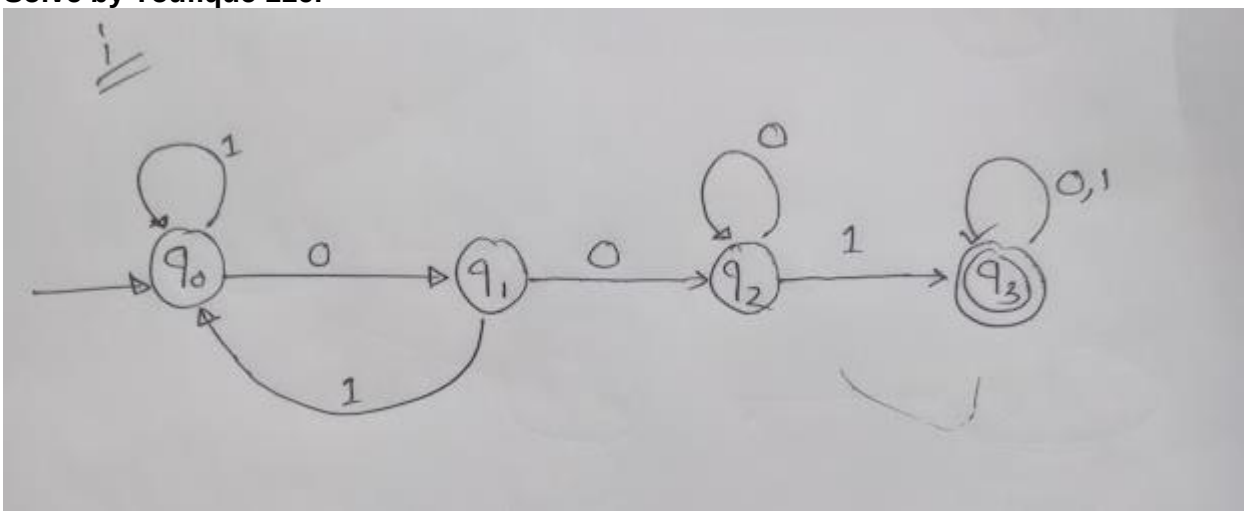
A.

Solve by Toufique 116:

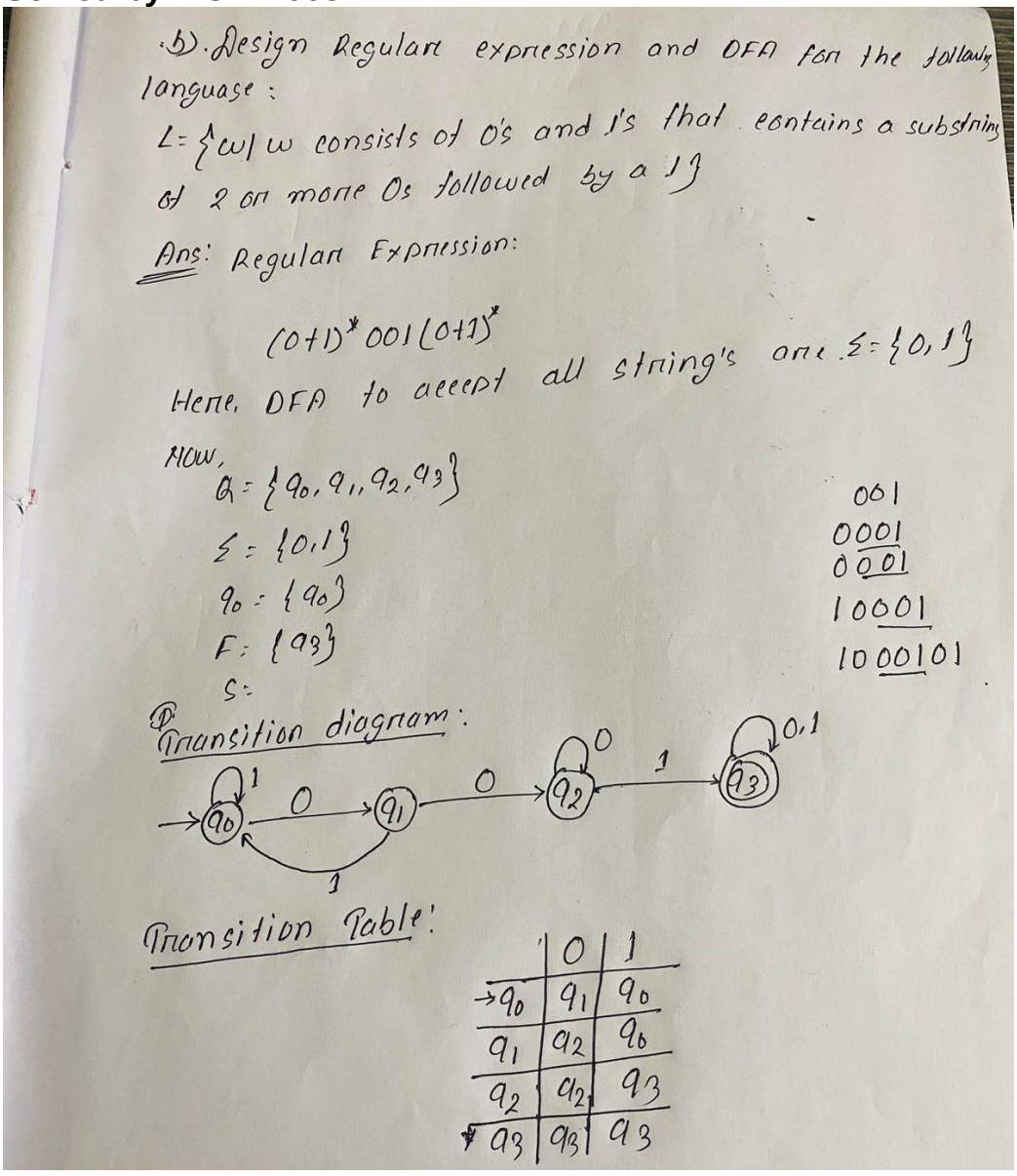


B.i.

Solve by Toufique 116:



Solved by Aishik-003



(b). Design Regular expression and DFA for the following language:

$L = \{w \mid w$ consists of 0's and 1's that contains a substring of 2 or more 0s followed by a 1$\}$

Ans: Regular Expression:

$$(0+1)^*001(0+1)^*$$

Here, DFA to accept all string's are. $\Sigma = \{0, 1\}$

Now,

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

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

$q_0 = \{q_0\}$

$F = \{q_3\}$

$\delta =$

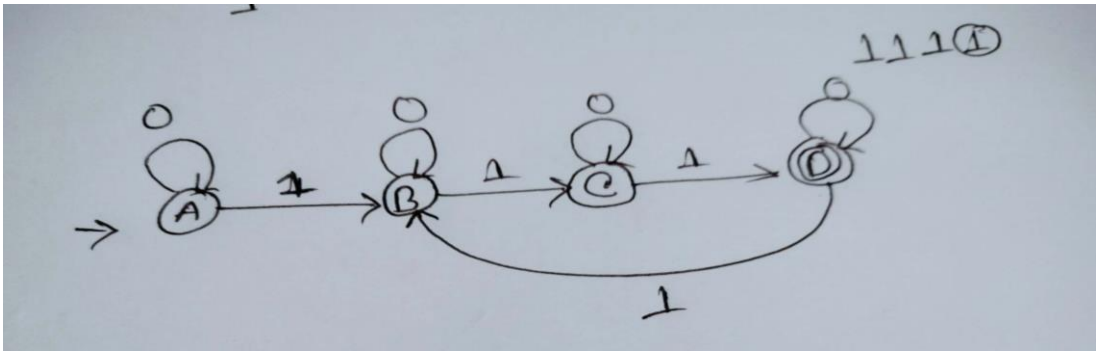
001
0001
00001
10001
1000101

Transition diagram:

Transition Table:

	0	1
→ q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_3
* q_3	q_3	q_3

B.ii.



Question 3. [Marks: 14]

a) Propose Regular Expressions for the following languages: [6]

i. L = {w | w consists of As and Bs in which two Bs do not come together.}
ii. L = {w | w consists of 0s and 1s which starts and ends with opposite symbol}
iii. L = {0, 000, 011, 010, 001, 01011, 00110, …}
iv. L = {00, 100, 0010, 1001, 110001, 1110000, …}

Ans: **solved by Younus-131**

(i) : (A + BA)* (B + ε) - **Explanation :** https://youtube.com/watch?v=IswH4GqO8Sg&feature=share

alternative : (B + ε) (A + AB)* ***Both are correct***

(ii): (0 (0|1)* 1) | (1 (0|1)* 0)

(iii): 0 ((1|0) (1|0))*
Language: "starts with 0 then any combination of 0's and 1's and the length of the string will be odd"

(iv): 1* 00 (0|1)*
Language: "starts with N number of 1's and 00 substring then any combination of 0's and 1's, N>=0"

b) Consider the grammar, G and answer the following questions: [4+2+2]

$S \rightarrow AcBd$
$A \rightarrow bA \mid \varepsilon$
$B \rightarrow cBd \mid a$

i. Find FIRST and FOLLOW sets.
ii. Construct the LL(1) parsing table.
iii. Show the moves a predictive parser makes on input "ccad".

Ans:

Ques: 3 (b)

$S \rightarrow AeBd$

$A \rightarrow bA \mid \epsilon$

$B \rightarrow eBd \mid a$

i. First (S) = {b, e} | FOLLOW (S) = {$}

First (A) = {b, ϵ} | FOLLOW (A) = {e}

FIRST (B) = {e, a} | FOLLOW (B) = {d}

FIRST (e) = {e}

FIRST (d) = {d}

FIRST (a) = {a}

ii

NON TERMINAL	INPUT SYMBOL				
	$	a	b	e	d
S			S→AeBd	S→AeBd	
A			A→bA	A→ϵ	
B		B→a		B→eBd	

[Toufique]

iii

MATCHED	STACK	INPUT	ACTION
	S $	eead$	
	AeBd $	eead $	output S→AeBd
	eBd $	eead $	" A→ϵ
e	Bd$	ead$	match e
e	eBdd $	ead $	~~match~~ B→eBd
ee	Bdd$	ad$	match e
ee	add$	ad$	B→a
eea	dd $	d$	match a
eead	d$	$	match d

Question 4. [Marks: 14]

a) Write down the Three Address Code for the following code snippet. [7]

```
a=4;
b=2;
cnt=0;
if(a<5)
    for( ; ; a--)
    {
      if(a==b)
      {
        b=a+b;
        cnt++;
        break;
      }
      cnt+=1;

    }
else
    switch(cnt)
    {
      case 1: a=1;
      case 2: a=2;
      case 3: a=3;
    }
```

Ans: [Toufique]

a = 4
b = 2
cnt = 0
if a < 5 goto **if_outer**
goto **else_block**

if_outer: if a == b goto **if_inner**
t1 = cnt + 1
cnt = t1
t4 = a - 1
a = t4
goto **if_outer**
if_inner: t2 = a + b
b = t2
t3 = cnt + 1
cnt = t3
goto **end_block**

else_block: if cnt == 1 goto **c1_block**
if cnt == 2 goto **c2_block**
if cnt == 3 goto **c3_block**
goto **end_block**

c1_block: a = 1

c2_block: a = 2
c3_block: a = 3
end_block:

b) Consider the following C/C++ code snippet. [7]

```
01: for (; ;) /*loop statement*/
02:  if (x == y) //conditional statement
03:    z = z+1;
04:  else
05:    z = z-(++a);
```

Describe the processes in the Lexical Analysis phase of a compiler and write down the output of each process for the given input. Use the following token classes, if needed.

WS:: '\n', '\t'
DEL:: ';'
KEY:: 'if ', 'else', 'for'
BOPS:: '+', '-', '*', '/'
UOPS:: '++', '--'
TEST:: '= ='
ASGN:: '='
ID:: 'a' – 'z'
NUM:: '-999' – '999'
PAR:: '(' , ')'

Ans:
[Toufique]

- Sometimes, lexical analyzers are divided into a cascade of two processes:
  - *Scanning*, consists of the simple processes that do not require tokenization of the input, such as deletion of comments and compaction consecutive white space characters into one.
  - *Lexical Analysis*, is the more complex portion, which produces tokens from the output of the scanner.

Scanning:
for (; ;) if (x == y) z = z+1; else z = z-(++a)
Lexical Analysis:
[KEY for] [PAR ( ] [DEL ;] [DEL ;] [PAR)] [KEY if] [PAR ( ] [ID x] [TEST ==] [ID y] [PAR)] [ID z] [ASGN =] [ID z] [BOPS +] [NUM 1] [DEL ;] [KEY else] [ID z] [ASGN =] [ID z] [BOPS -] [PAR ( ] [UOPS ++] [ID a] [PAR)] [DEL ;]

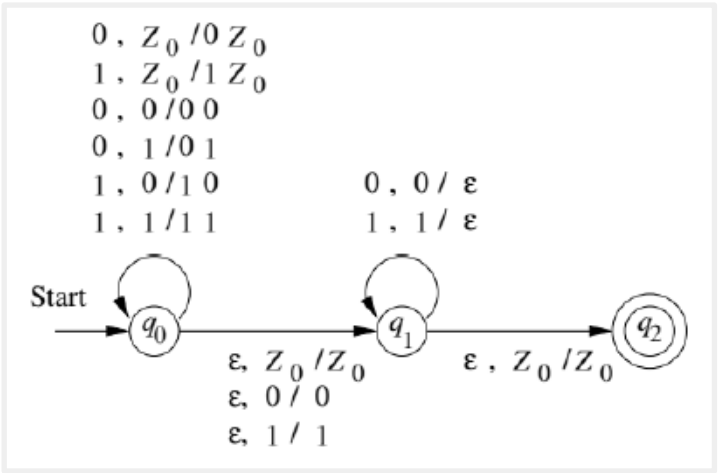
tion 5. [Marks: 14]

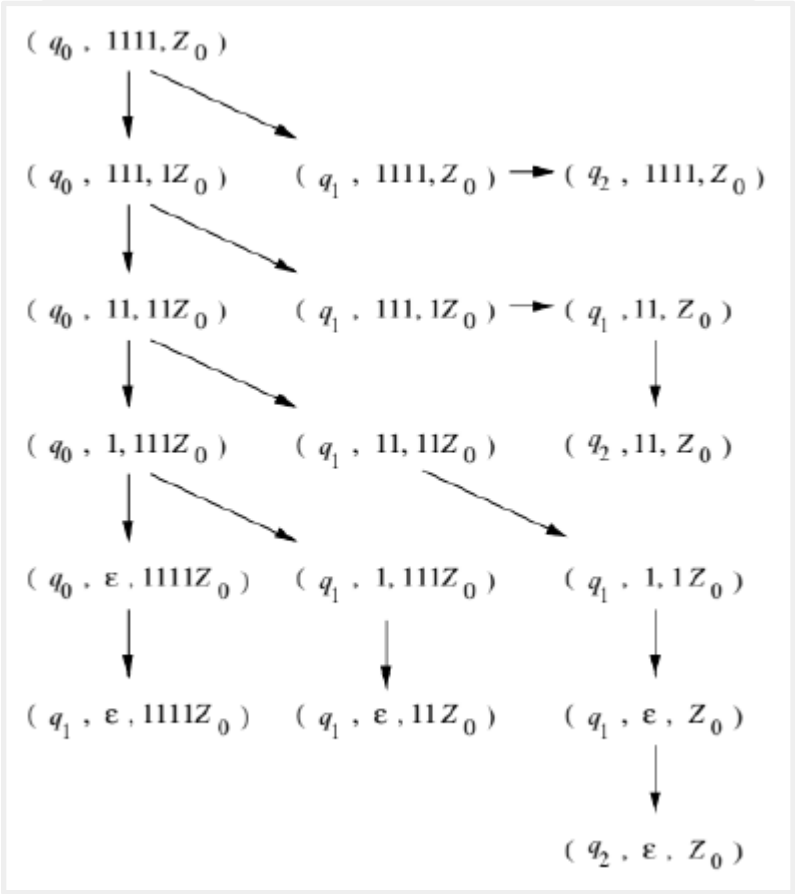
a) Design a PDA to accept the language, L_{wwr} which is the even-length palindromes over alphabet {0, 1} and show the ID's of the PDA on input 1111. [7]

b) Construct the LR(0) automaton for the following grammar: [7]

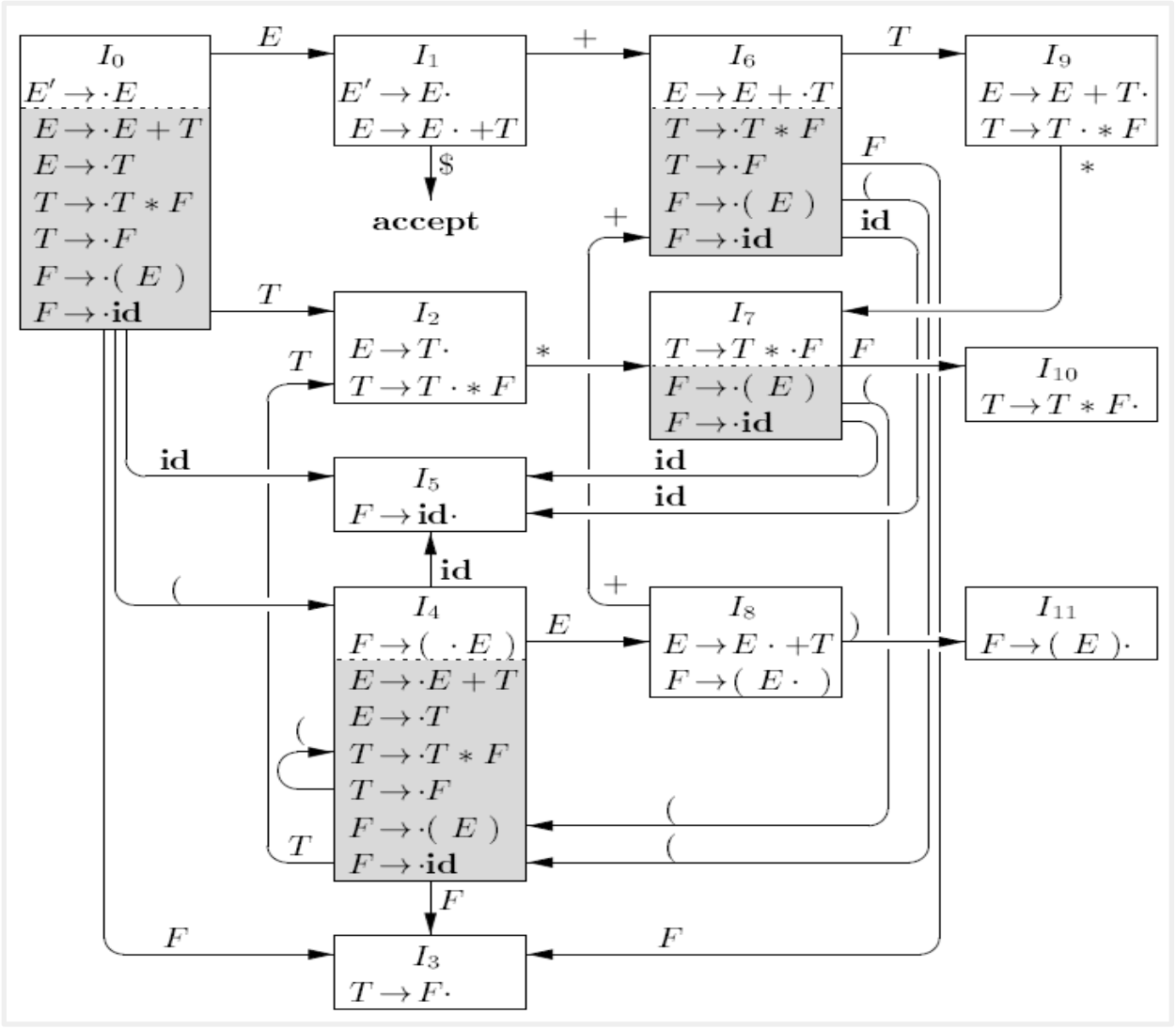
$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid id$$

5.a.





5.b.



Question 6. [Marks: 14]

a) Write a Context Free Grammar to describe the language of palindromes over alphabet {0, 1} and show the leftmost derivation of the string **0110110** with proper annotation. [7]

Ans:

P → epsilon | 0 | 1 | 0 P 0 | 1 P 1

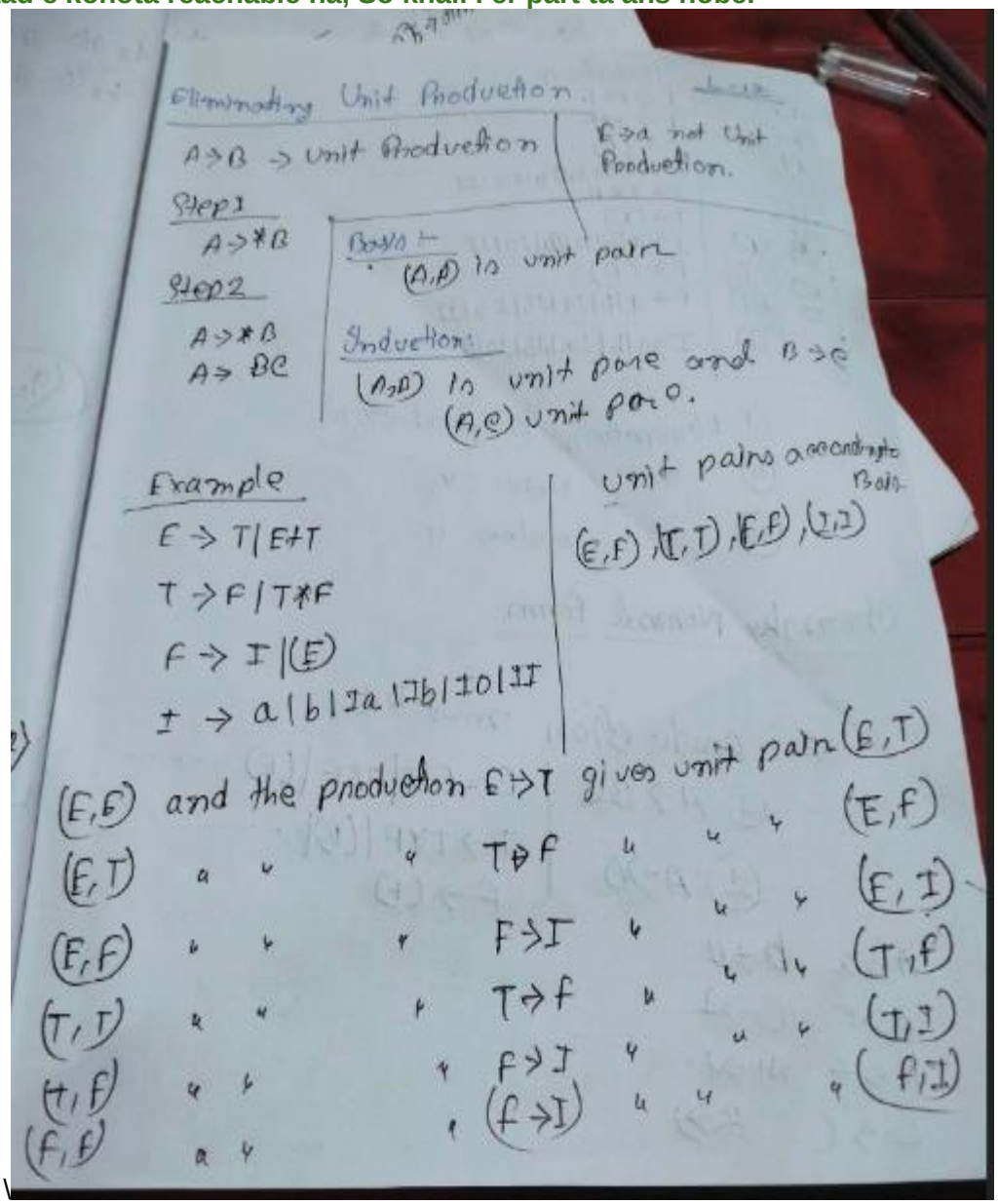
P
0 P 0
0 1 P 1 0
0 1 1 P 1 1 0
0 1 1 0 1 1 0

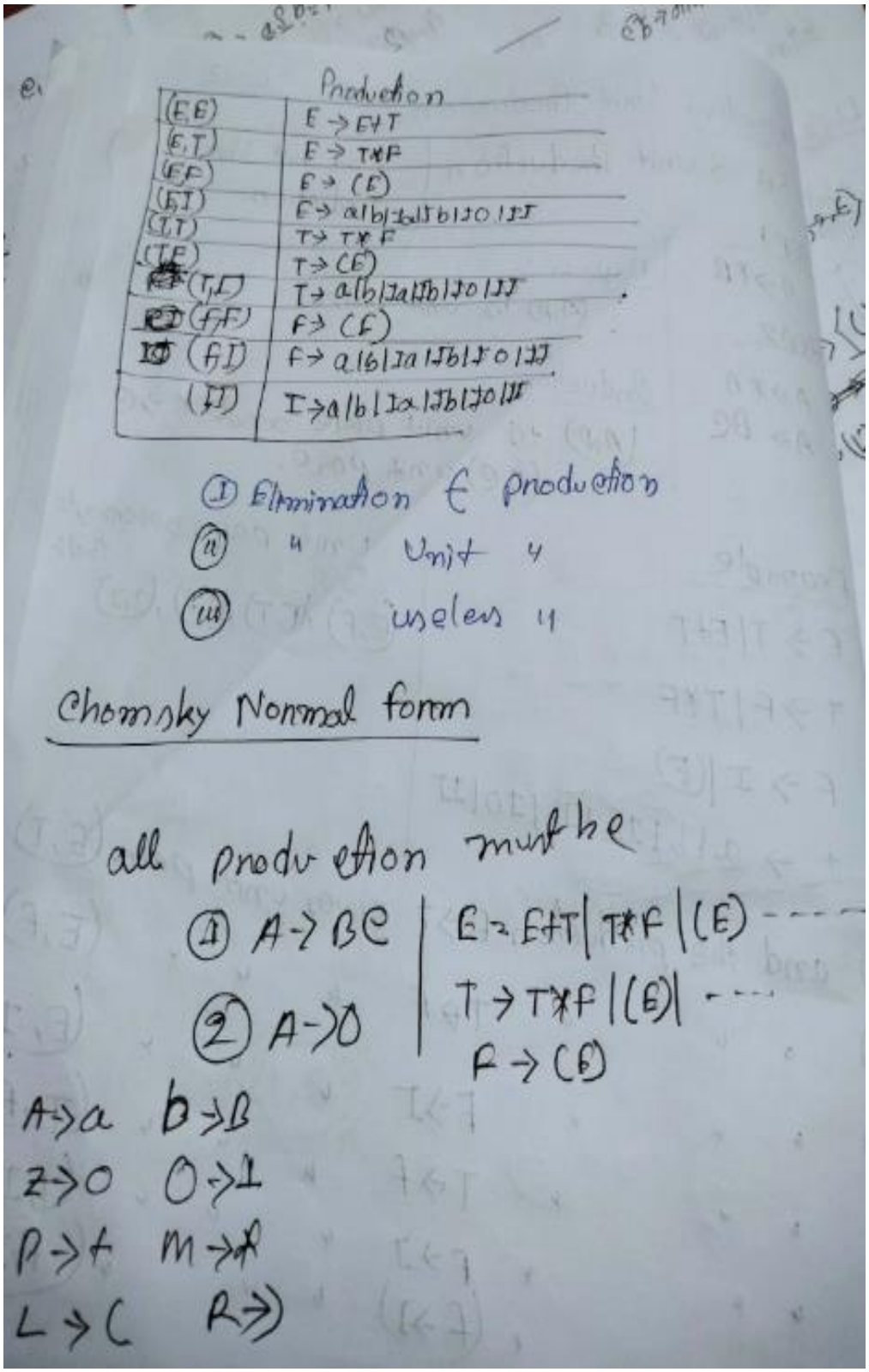
b) Consider the grammar and answer following questions: [4+3]

$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$
$$F \rightarrow I \mid (E)$$
$$T \rightarrow F \mid T * F$$
$$E \rightarrow T \mid E + T$$

i. Remove Unit Productions from the above grammar.
ii. Convert the resulting grammar to Chomsky Normal Form.

Ans: **Start symbol jodi I hoy, Chomsky er form e anar time e useless symbol eliminate korte hobe. I er production baad e konota reachable na, So khali I er part ta ans hobe.**





	Production
(E,E)	E → E+T
(E,T)	E → T*F
(E,F)	E → (E)
(E,I)	E → a|b|Ia|Ib|I0|I1
(T,T)	T → T*F
(T,F)	T → (E)
(T,I)	T → a|b|Ia|Ib|I0|I1
(F,F)	F → (E)
(F,I)	F → a|b|Ia|Ib|I0|I1
(I,I)	I → a|b|Ia|Ib|I0|I1

(i) Elimination ∈ production

(ii) " Unit "

(iii) useless "

Chomsky Normal form

all production must be

(1) A → BC

(2) A → a

E → E+T | T*F | (E) ---

T → T*F | (E) | ---

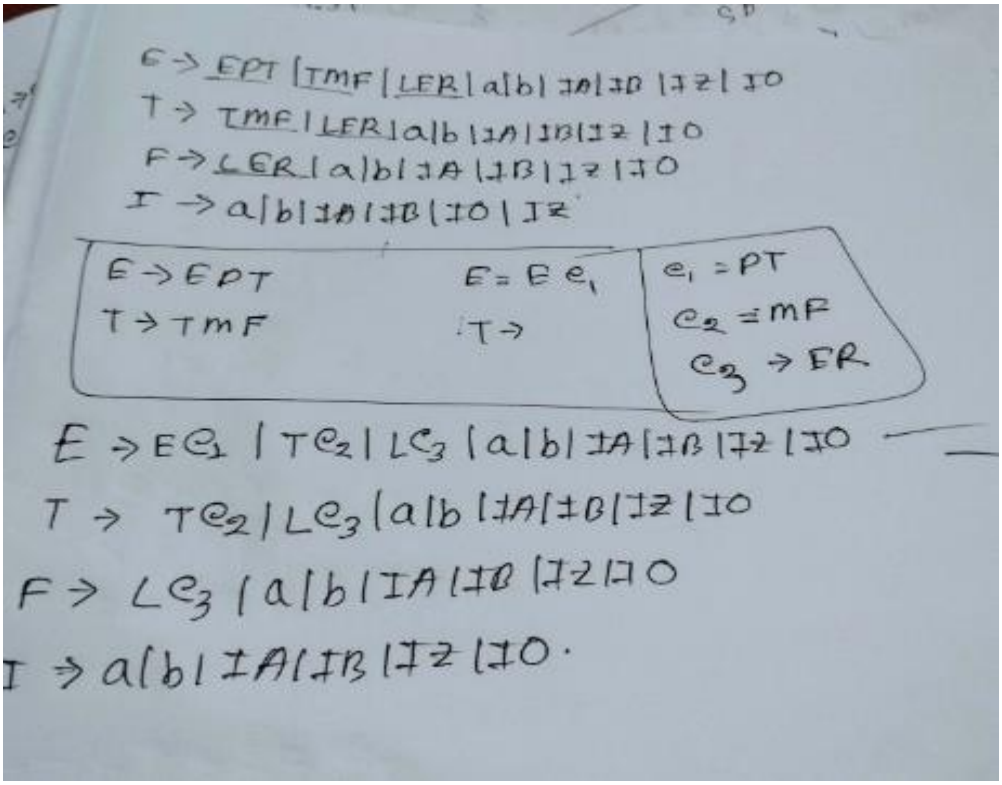
F → (E)

A → a b → B

Z → 0 0 → 1

P → + M → *

L → (R →)



[Toufique] [**I ke start symbol dhore kora**]

(F, F) and the production F -> I gives us unit pair (F, I)
(T, T) and the production T -> F gives us unit pair (T, F)
(T, F) and the production F -> I gives us unit pair (T, I)
(E, E) and the production E -> T gives us unit pair (E, T)
(E, T) and the production T -> F gives us unit pair (E, F)
(E, F) and the production F -> I gives us unit pair (E, I)

Pair	Productions
I, I	I -> a | b | Ia | Ib | I0 | I1
F, F	F -> (E)
F, I	F -> a | b | Ia | Ib | I0 | I1
T, T	T -> T * F
T, F	T -> (E)
T, I	T -> a | b | Ia | Ib | I0 | I1
E, E	E -> E + T
E, T	E -> T * F
E, F	E -> (E)
E, I	E -> a | b | Ia | Ib | I0 | I1

After removing unit productions we get,
I -> a | b | Ia | Ib | I0 | I1
F -> (E) | a | b | Ia | Ib | I0 | I1
T -> T * F | (E) | a | b | Ia | Ib | I0 | I1
E -> E + T | T * F | (E) | a | b | Ia | Ib | I0 | I1

A grammar G is said to be in Chomsky Normal Form (CNF) if it has no epsilon productions, unit productions or useless symbols and all the productions of the G are in one of two simple forms, either:
1. A -> BC, where A, B and C, are each variable or
2. A-> a where A is a variable and a terminal

The grammar doesn't have any epsilon productions and the resulting grammar of (i) does not contain any unit production.
Now we have to remove useless productions.

F, T, E are not reachable as there is no such derivation from starting symbol I where I =>* αXβ where X = F, T, E
Eliminating these we get,
I -> a | b | Ia | Ib | I0 | I1
All of these productions are generating as there exist derivation X =>* w where w is terminal string for every production.

Let, A -> a
B -> b
0 -> Z
1 -> O

Using these productions we get,
I -> a | b | IA | IB | IZ | IO

Question 7. [Marks: 14]

a) How can you compute the set of Generating Symbols and the set of Reachable Symbols of a grammar? Write the basis and the induction for both the cases. [4]

b) Write short note on Epsilon-Closure of a state of ε-NFA and illustrate how NFA is different from DFA with appropriate examples. [5]

c) What do you mean by "powers of an alphabet"? Describe the roles of Regular Expression and Deterministic Finite Automata in compiler design. [5]

A.

- A symbol X is *generating* if $X \Rightarrow^* w$ for some terminal string w. Note that every terminal is generating, since w can be that terminal itself, which is derived by zero steps.
- A symbol X is *reachable* if there is a derivation $S \Rightarrow^* \alpha X \beta$ for some α and β.

Consider the grammar:

$$S \rightarrow AB \mid a$$
$$A \rightarrow b$$

Eliminating non-generating symbols: All symbols but B are generating. If we eliminate B, we must eliminate the production $S \rightarrow AB$, leaving the grammar:

$$S \rightarrow a$$
$$A \rightarrow b$$

Eliminating non-reachable symbols: Only S and a are reachable from S. Eliminating A and b leaves only the production:

$$S \rightarrow a$$

Let G = (V, T, P, S) be a grammar. To compute the generating symbols of G, the following induction is performed.

Basis: Every symbol of T is obviously generating; it generates itself.
Induction: Suppose there is a production A → α, and every symbol of α is already known to be generating; then A is generating.

To compute the reachable symbols of G, the following induction is performed.

Basis: S is surely reachable.
Induction: Suppose we have discovered that some variable A is reachable. Then for all productions with A in the head, all the symbols of the bodies of those productions are also reachable.

B.

C.- by 24

Powers of an Alphabet:

- If Σ is an alphabet, the set of all strings of a certain length from that alphabet can be expressed by using an exponential notation.
- Σ^k is defined as the set of strings of length k, each of whose symbols is in Σ.
- $\Sigma^0 = \varepsilon$, no matter what the alphabet Σ is. In other words, ε is the only string of length 0.
- Example: If Σ = {a, b, c} then Σ^1 = {a, b, c}, Σ^2 = {aa, ab, ac, ba, bb, bc, ca, cb, cc}, Σ^3 = {aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc}.

N.B: Que er 2nd part baki

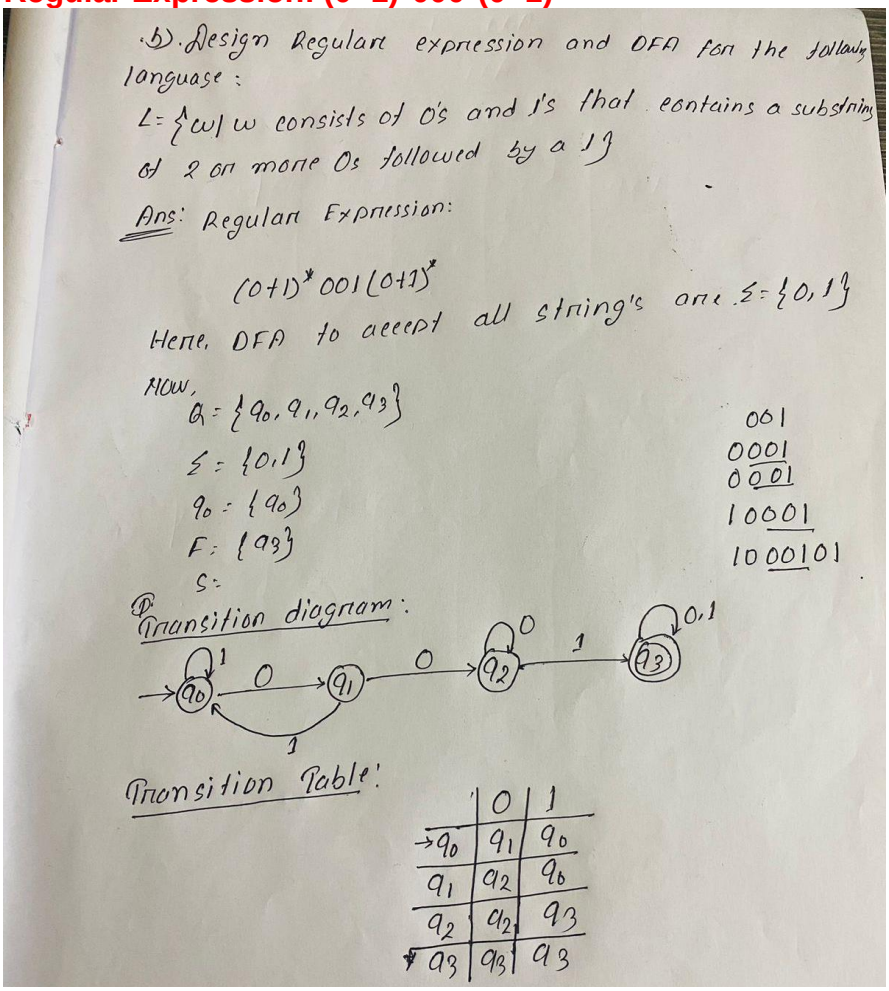
Recursive-40 Final

1. a) What do you understand by Formal Languages? Explain the relationship between Formal Languages and Compilers. Provide examples where necessary.

Ans:

b) Design Regular Expression and DFA for the following language: L = {w | w consists of 0's and 1's that contains a substring of 2 or more 0s followed by a 1}

Regular Expression: (0+1)*000*(0+1)*

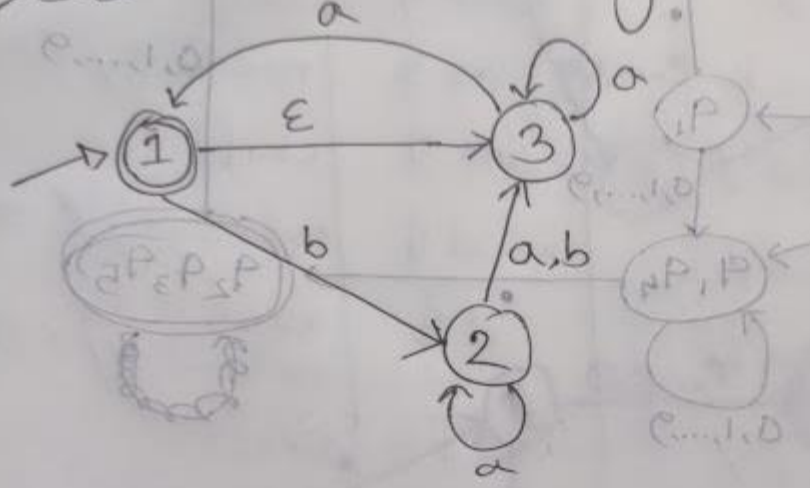


c) Convert the following ε-NFA to a DFA.

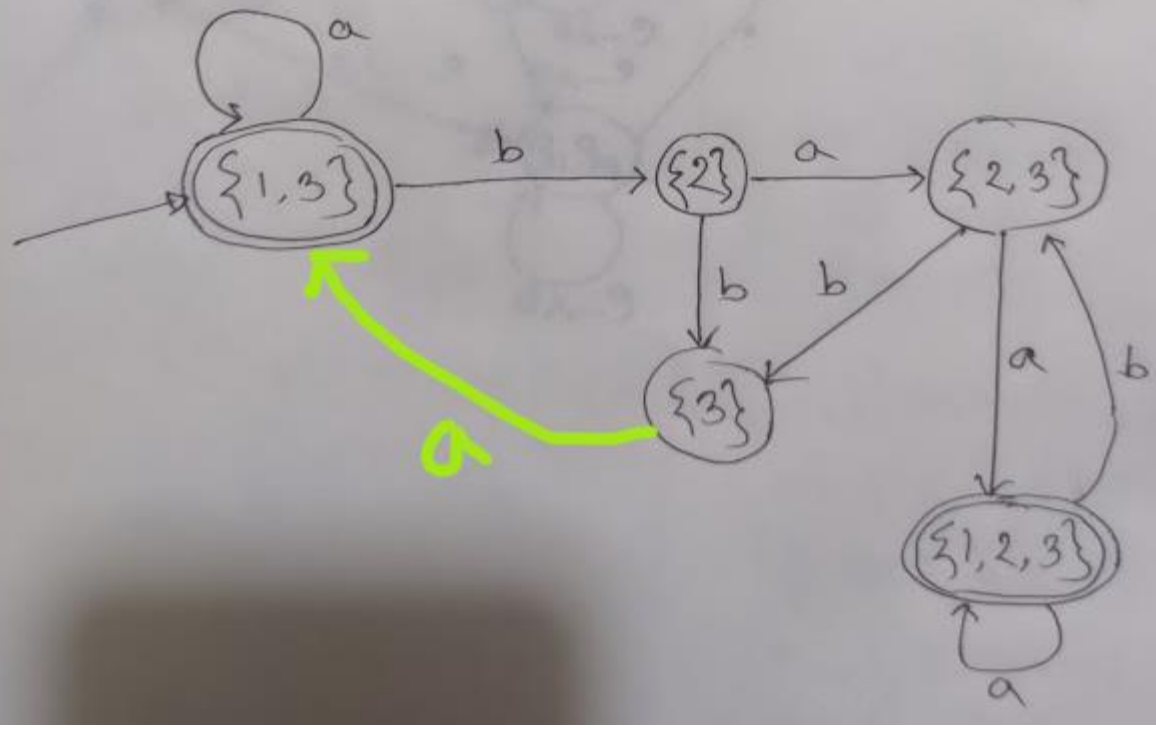
	ε	a	b
→*1	{3}	∅	{2}
2	∅	{2, 3}	{3}
3	∅	{1, 3}	∅

Ans:

Transition diagram for ε-NFA:



equivalent DFA:



Question 2. [Marks: 12.5]

a) Briefly describe the preliminary simplifications which are required to get the Chomsky Normal Form of a CFG. **[3.5]**

b) What do you mean by ambiguous grammars? Demonstrate that the following grammar is ambiguous for the expression **a + bc**. **[1+3]**

$$R \rightarrow R + R \mid RR \mid R^* \mid a \mid b \mid c$$

c) Write down the Three Address Code for the following code snippet. **[5]**

```
ch=1;
if(ch==1)
   for(j=1; j<=10; j++)
      ch=ch+j;
else
   if(ch>1)
   {
      i=10;
      while(i>0)
         i--;
   }
   else
      ch=0;
```

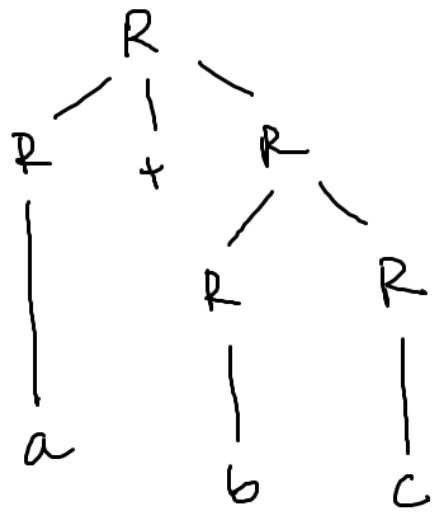
2.a. Tamal 122
3 steps:
1. Eliminate epsilon production
2. Eliminate unit production
3. Eliminate useless symbol

2.b. [Ambiguous Grammar - YouTube]

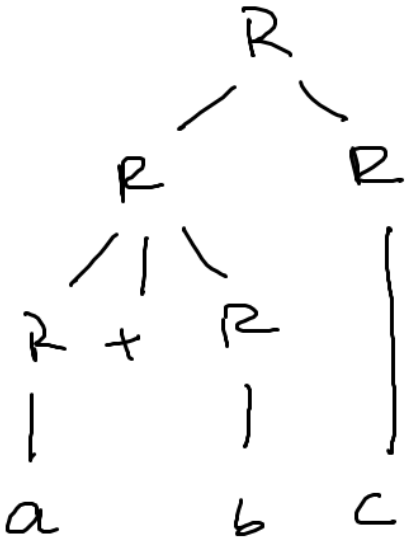
Ambiguous Grammar

- We assume that a grammar uniquely determines a structure for each string in its language.
- However, we shall see that not every grammar does provide unique structures.
- A grammar is called *ambiguous* when it fails to provide unique structure for each string in its language.

[Sohom]



a+bc



a+bc

2.c. **Solved by Younus-131**

```
Ch = 1
if ch == 1 goto A
goto B

A: j = 1
C: if j<=10 goto D
   goto Z

D: t1 = ch + j
   ch = t1
   t1 = j + 1
   j = t1
   goto C

B: if ch > 1 goto E
   goto F

E: i = 10
G: if i > 0 goto H
   goto Z

H: t1 = i - 1
   i = t1
   goto G

F: ch = 0

Z: End
```

Question 3. [Marks: 12.5]

a) How Syntax Analysis and Semantic Analysis of a compiler are different from each other? Write down the importance of Error Recovery during parsing. **[2+1.5]**

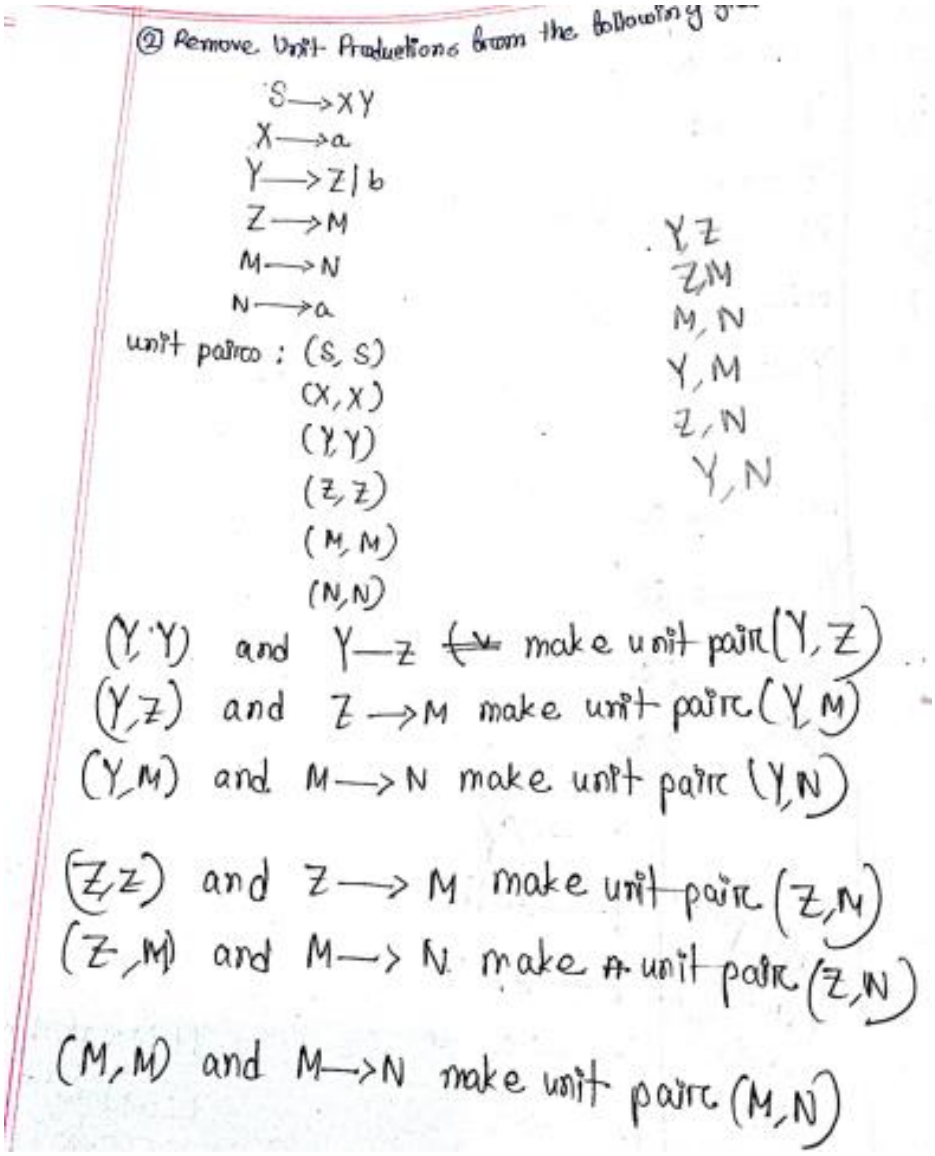
b) Remove Unit Productions from the following grammar. **[4]**

$$S \rightarrow XY$$
$$X \rightarrow a$$
$$Y \rightarrow Z \mid b$$
$$Z \rightarrow M$$
$$M \rightarrow N$$
$$N \rightarrow a$$

c) Design an NFA that accepts all strings over {0, 1} where the 3rd symbol from the end is a 1 and show the states the NFA is in during the processing of input sequence **01100**. **[5]**

3.a.

3.b.



Pairs	Productions
(S,S)	S → XY
(X,X)	X → a
(Y,Y)	Y → b
(Z,Z)	
(M,M)	
(N,N)	N → a
(Y,Z)	
(Y,M)	
(Y,N)	Y → a
(Z,M)	
(Z,N)	Z → a
(M,N)	M → a

so the final grammar becomes.

S → XY

X → a

Y → a | b

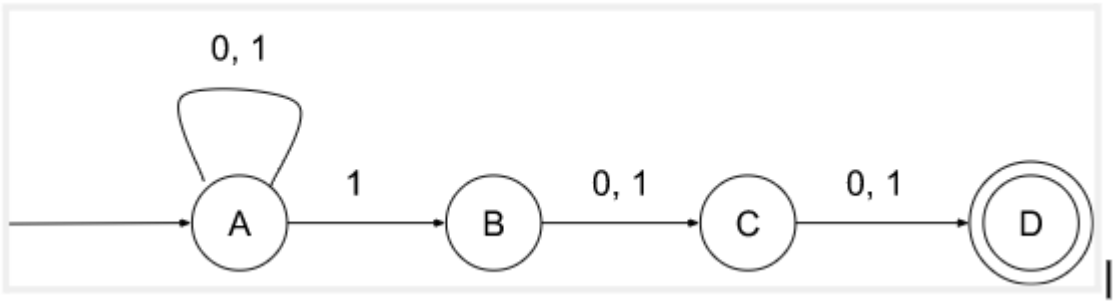
Z → a

M → a

N → a

3.c.

Ans: *by 24*



Answer baki

Question 4. [Marks: 12.5]

a) How left recursive grammar is problematic for top-down parsing? Eliminate left recursion from the following grammar. **[1.5+2]**

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L\,,\, S \mid S$$

b) Consider the following grammar for infix expressions.

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow \text{digit}$$

i. Write appropriate Semantic Rules to solve the problem of generating prefix equivalents of infix expressions. **[2]**

ii. Find the Annotated Parse Tree for the expression **1+3+2*5** according to your semantic rules. **[2]**

c) Draw the transition diagram of the following DFA and minimize it. **[5]**

	a	b
→ 0	2	1
*1	1	2
*2	2	2

4.a. Left recursive grammars, such as G, are unsuitable for recursive-descent parsing because a left-recursive production leads to an infinite recursion.

S→(L)|a

L→ SL'

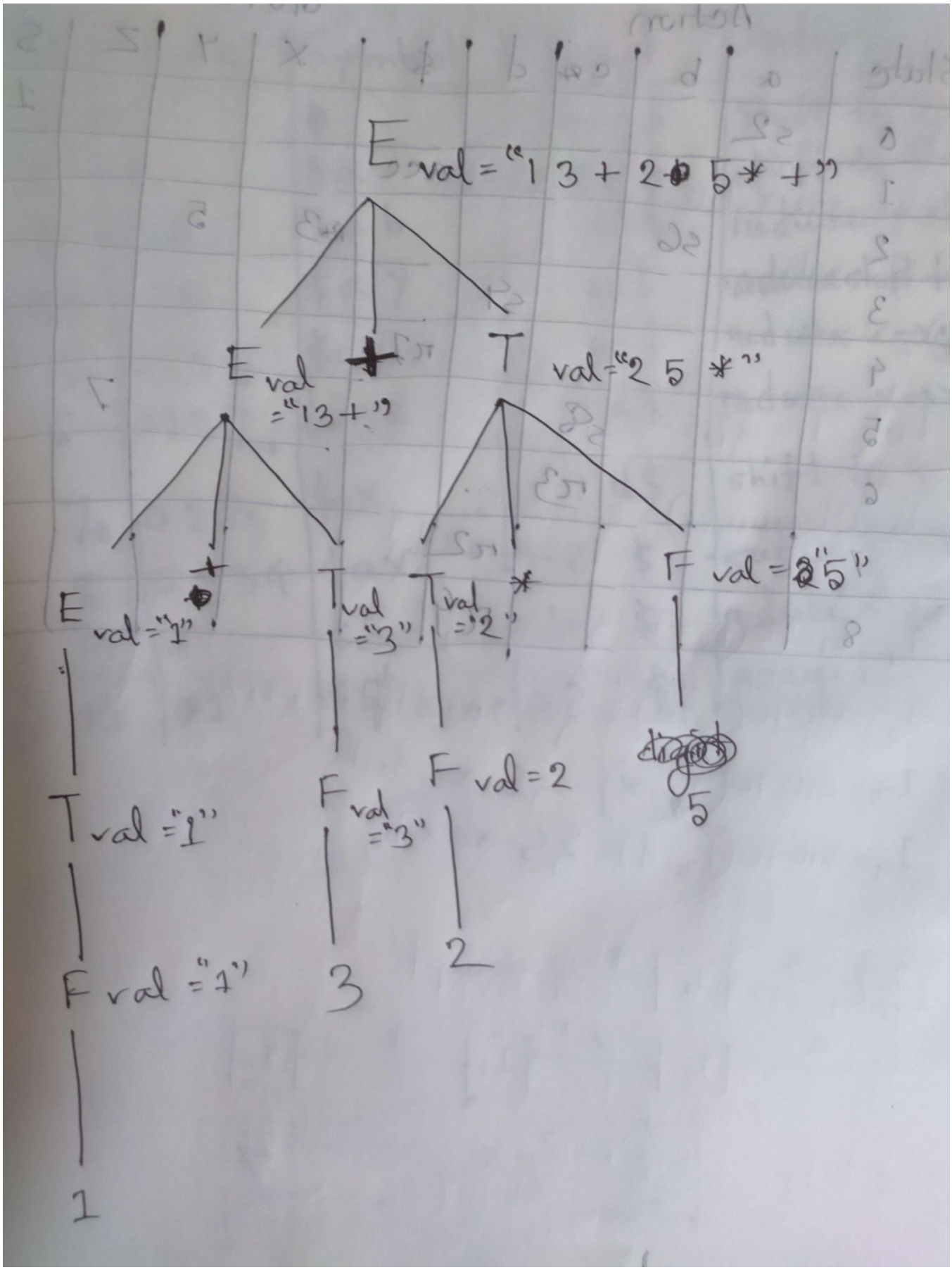
L'→,SL'|epsilon

4.b.
[Toufique] [Not sure]

i.

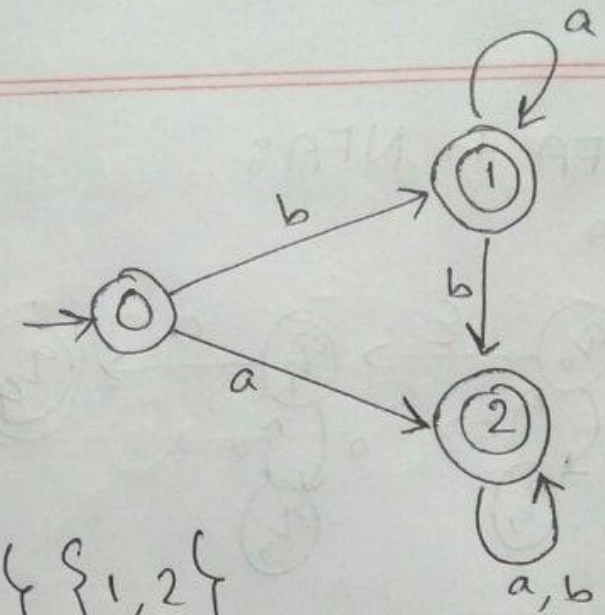
Production	Semantic Rules
E --> E_1 + T	{E.val = E_{1}.val || T.val || '+'}
E --> T	{E.val = T.val}
T --> T_1 * F	{T.val = T_{1}.val || F.val || '*'}
T --> F	{T.val = F.val}
F --> digit	{F.val = digit}

ii.


E val = "1 3 + 2 5 * +"
E val = "13+"
+
T
val = "2 5 *"
E val = "1"
+
T val = "3"
T val = "2"
*
F val = "5"
T val = "1"
F val = "3"
F val = 2
F val = "1"
1
3
2
5

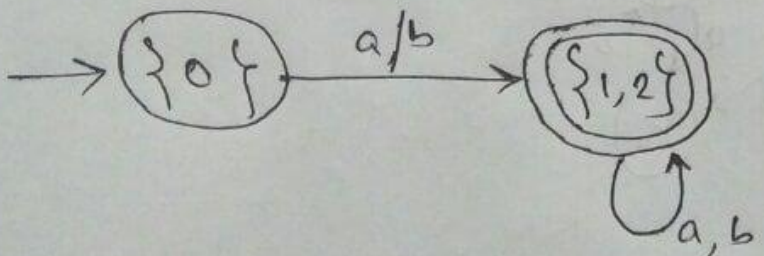
4.c. **Solved by Younus-131**

	a	b
→0	2	1
*1	1	2
*2	2	2



0 Equivalence : {0} {1, 2}

1 Equivalence : {0} {1, 2}



Question 5. [Marks: 12.5]

a) Explain with the help of an example how lexical analyzer groups the stream of characters of a source program into lexemes. **[3.5]**

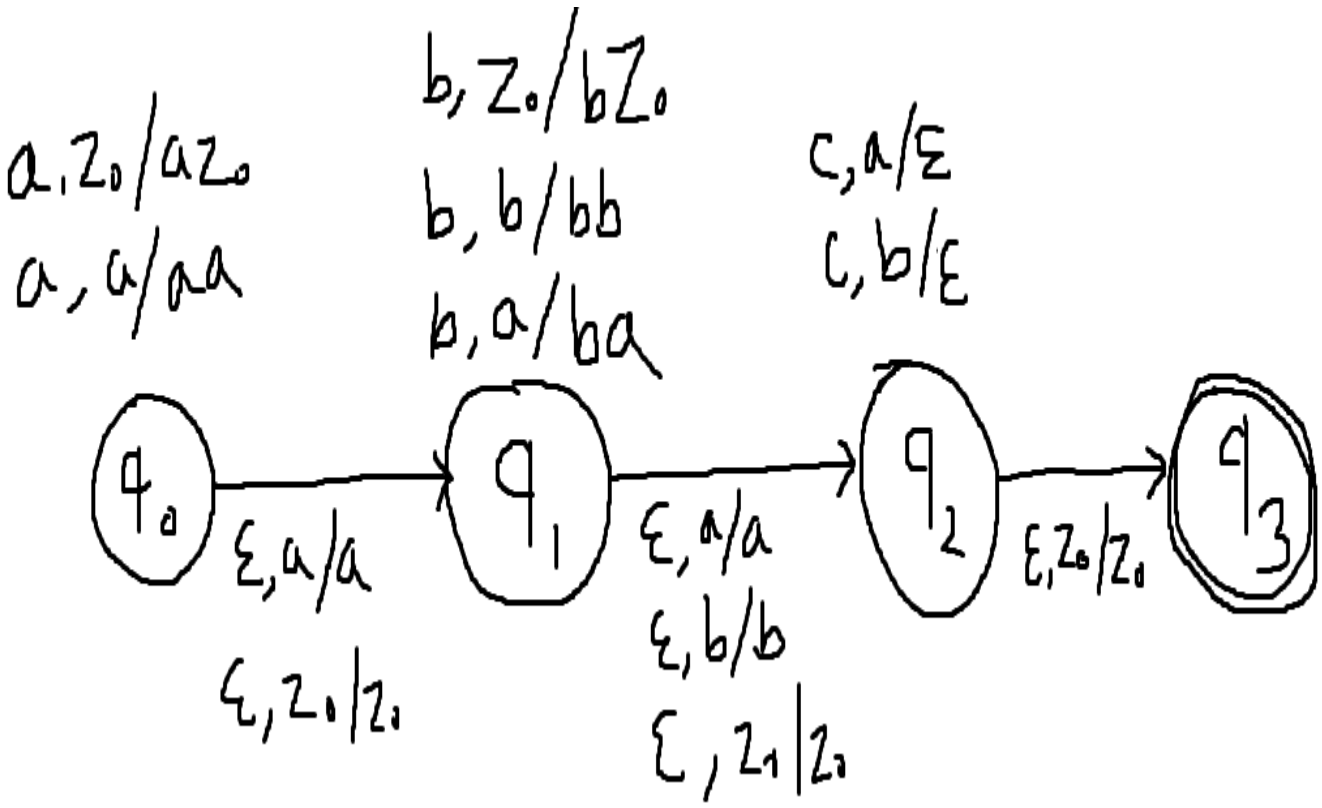
b) Define Item and Reduce/Reduce conflict. Is there any similarity between the Function GOTO and the transition function of finite state machines? Justify your answer. **[2+2]**

c) Design a PDA that accepts the language, $L = \{a^x b^y c^z \mid x, y, z \geq 0 \text{ and } x + y = z\}$ **[5]**

5.a.

5.b.

5.c.
[Toufique]



Question 6. [Marks: 12.5]

a) What is Parse Tree? Describe the relationship between Parsers and Context Free Grammars. **[1+2.5]**

b) Consider the following LL(1) parsing table and show the moves a top-down parser makes for the input string ***accd***. **[3]**

	a	b	c	d	$
S	S → aXd				
X		X → YZ	X → YZ	X → ε	
Y		Y → b	Y → ε	Y → ε	
Z			Z → cX	Z → ε	

c) Construct DFAs from the following regular expressions: **[3+3]**

i. 11*01*01*
ii. 1*01*0(0|1)*

6.a.

- A parse tree pictorially shows how the start symbol of a grammar derives a string in the language.
- From left to right, the leaves of a parse tree form the *yield* of the tree, which is the string generated or derived from the nonterminal at the root of the parse tree.
- Formally, given a context-free grammar, a parse tree according to the grammar is a tree with the following properties:
    - The root is labeled by the start symbol.
    - Each leaf is labeled by a terminal or by ε.
    - Each interior node is labeled by a nonterminal.
    - If A is the nonterminal labeling some interior node and X, Y, Z are the labels of the children of that node from left to right, then there must be a production $A \rightarrow XYZ$. Here, X, Y, Z each stand for a symbol that is either a terminal or nonterminal. As a special case, if $A \rightarrow \varepsilon$ is a production, then a node labeled A may have a single child labeled ε.

So, an interior node and its children correspond to a production; the interior node corresponds to the head of the production, the children to the body.

A Production in CFG

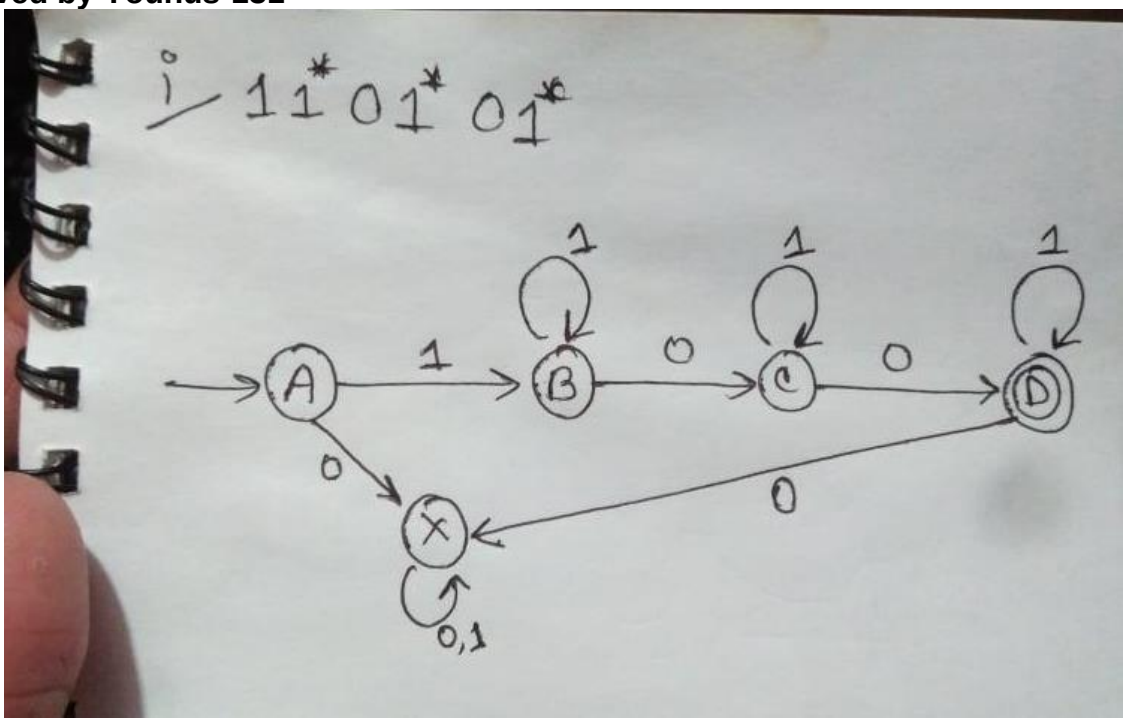
$A \rightarrow XYZ$

An Interior Node of a Parse Tree

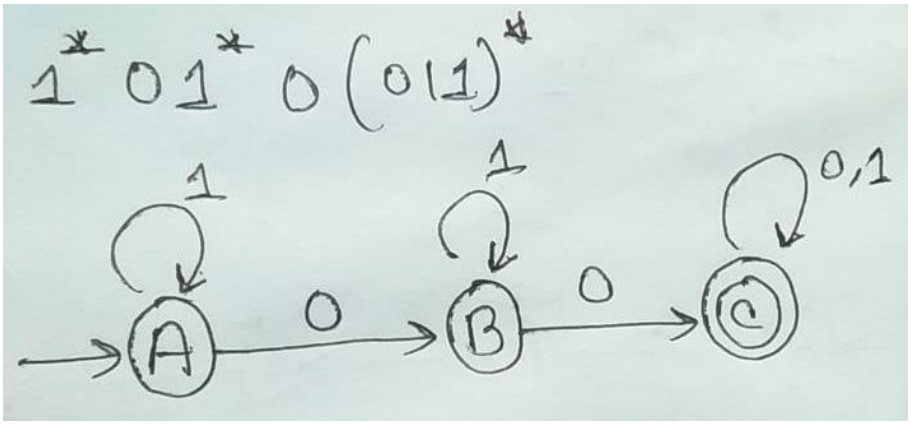
6.b.LL(1) Parsing e $ sign pore hy
Like S$
dXa$
And so on
And also string ee o akta $ sign hobe sob gulu te.

$S	accd	S>aXd
$dXa	accd	
$dX	ccd	X>YZ
$dZY	ccd	Y>e
$dZ	ccd	Z>cX
$dXc	ccd	
$dX	cd	X>YZ
$dXZY	cd	Y>e
$dXZ	cd	Z>cX
$dXX	c	cd
$dXX	d	X>e
$dX	d	X>e
$d	d	
$		

6.c. i. **Solved by Younus-131**



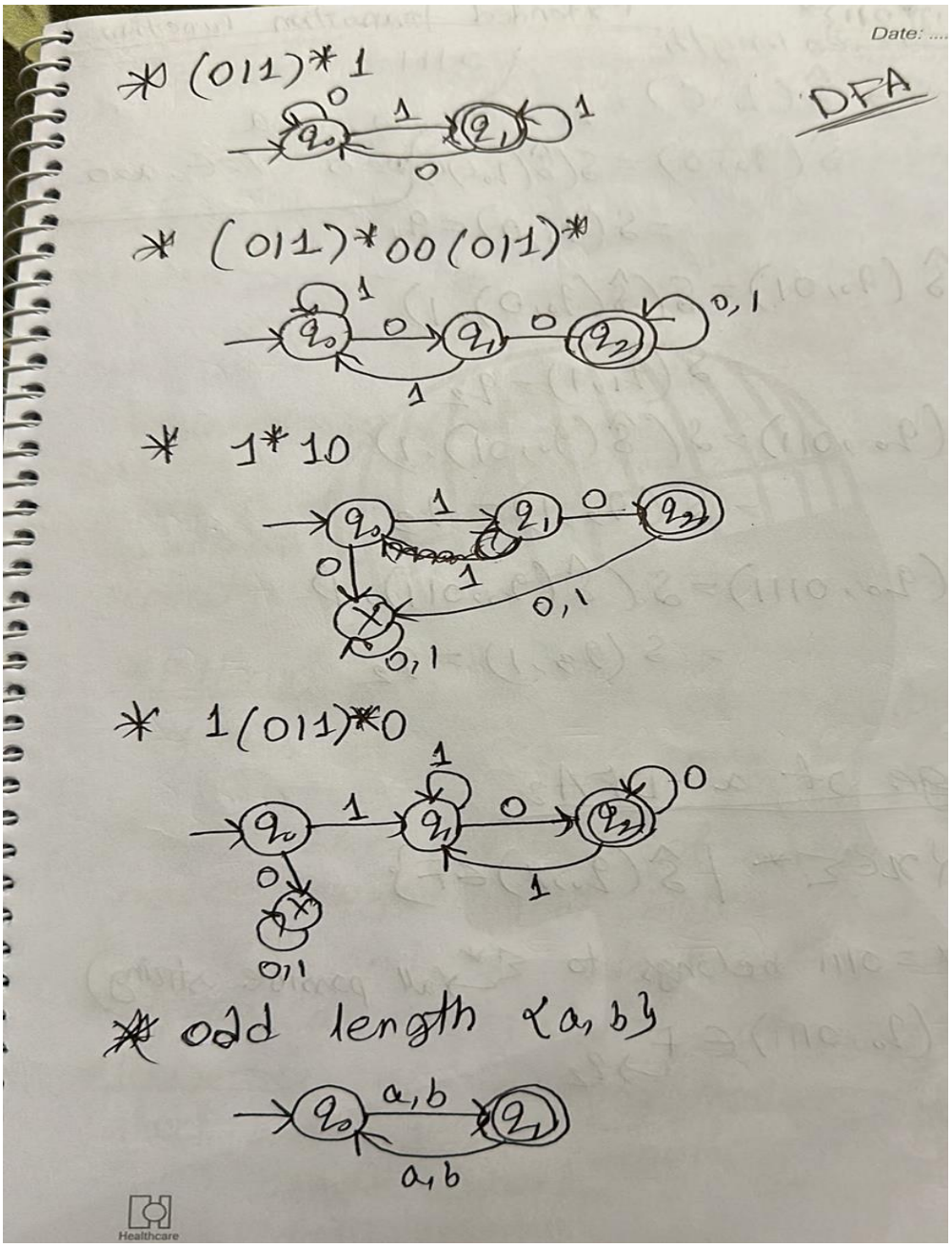
6.c. ii. **Solved by Younus-131**



EXERCISE

Exercise:

1. Construct DFAs for the regular languages represented by the regular expressions (0|1)*1, (0|1)*00(0|1)*, 1*10, 1(0|1)*0 over the alphabet {0, 1}.
2. Design a DFA that accepts strings of odd length over alphabet {a, b}.
3. Propose regular expressions and DFAs for the keyword *int* and any valid *C identifier*.



a) Describe Epsilon-Closure of a state in finite automata with an example.

b) Illustrate the concepts of Generating and Reachable symbols in your own words.

Answer :

c) What is the importance of DFA Minimization?

a) Duplicate declaration of an identifier in a source program can be detected with the help of a symbol table. Explain the process with an example.

b) Recursive-descent parsing may require backtracking to find correct production for a nonterminal. Explain a technique that can avoid the backtracking.

c) What is the configuration of a PDA? Design a PDA that accepts the language, $L = \{0^n 1^n 0^m \mid m, n \geq 1\}$.

a) What is Lexeme? Explain with the help of an example how lexical analyzer groups the stream of characters of a source program into lexemes. **[1+3]**

a) Intermediate representation of a source program bridges between two phases of a compiler. Write short notes on those two phases.

a) What is Item? Explain the significance of the dot (.) in an Item in your own words.

6. a) Define Compiler and draw the block diagram showing different components of a compiler. [3]

   b) Describe the Closure or Kleene Closure operation with an example and construct regular language from the regular expression **10*1.** [3]

   c) Design DFAs accepting the following strings over the alphabet {0, 1}: [4+4]
      i. The set of all strings that begin with 01 and end with 11.
      ii. The set of all strings such that the number of 1's is multiple of 3.

7. a) When a grammar is said to be Ambiguous? Explain with an example. [3]

   b) What are the purposes of Linker and Loader? Explain the function of the *analysis* phase of a compiler. [2+2]

3. a) How can you compute the set of Generating Symbols and the set of Reachable Symbols of a grammar? Write the basis and the induction for both the cases. [4]

b) What is the role of the function *match* in predictive parsing? Illustrate the concept of *ε-closure* of a state in a finite automaton with an example. [2+3]

4. a) Define Context Free Grammar (CFG) describing all of its components. Why are these grammars called context free? [3+1]

5. a) What is *Instantaneous Description* or *ID* of a Pushdown Automata (PDA)? Illustrate the constraints that turn a PDA to a DPDA. [4]

b) Recursive-descent parsing may require backtracking to find correct production for a nonterminal. Explain a technique that can avoid the backtracking. [4]

6. a) What do you know about Item of a grammar and the function GOTO in LR parsing? Answer using appropriate examples. [4]

7. a) What is Left Recursion? Describe the elimination process of left recursion with an example. [4]

b) Describe the structure of the LR parsing table. [4]